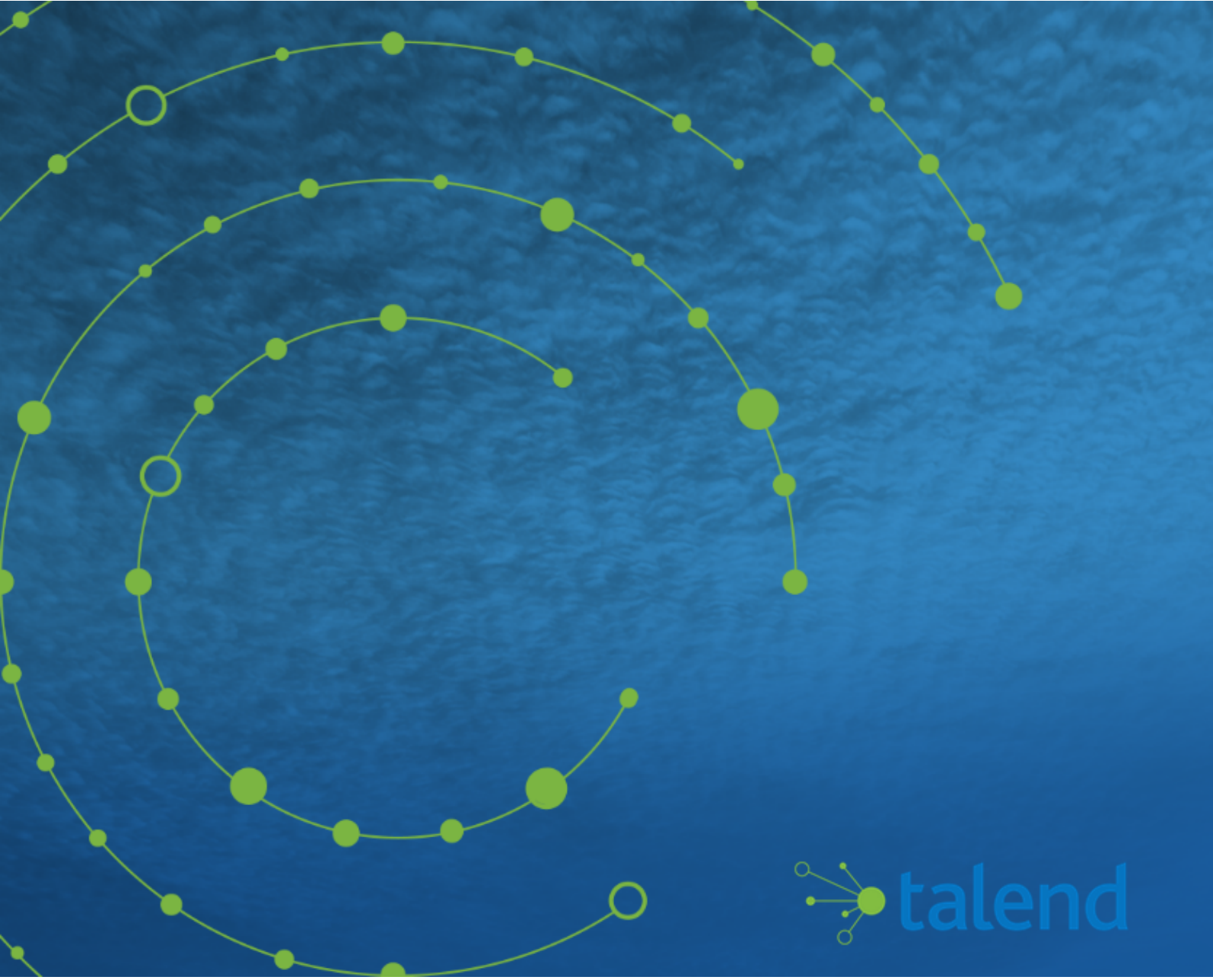

Talend ESB Mediation Developer Guide

7.1.1

Contents

Copyleft

Adapted for 7.1.1. Supersedes previous releases.

Publication date: October 15, 2019

The content of this document is correct at the time of publication.

However, more recent updates may be available in the online version that can be found on Talend Help Center.

Notices

License Agreement

JSON, JSR 305: Annotations for Software Defect Detection in Java, JUnit, Jackson Java JSON-processor, Java API for RESTful Services, Jaxb, Jaxen, Jettison, Jetty, Joda-Time, Json Simple, MetaStuff, Mondrian, OpenSAML, Paraccel JDBC Driver, PostgreSQL JDBC Driver, Resty: A simple HTTP REST client for Java, Rocoto, SL4J: Simple Logging Facade for Java, SQLite JDBC Driver, Simple API for CSS, SshJ, StAX API, StAXON - JSON via StAX, Talend Camel Dependencies (Talend), The Castor Project, The Legion of the Bouncy Castle, W3C, Woden, Woodstox : High-performance XML processor, XML Pull Parser (XPP), Xalan-J, Xerces2, XmlBeans, XmlSchema Core, Xmlsec - Apache Santuario, Zip4J, atinject, dropbox-sdk-java: Java library for the Dropbox Core API, google-guice. Licensed under their respective license.

Introduction to Apache Camel with Talend ESB

Talend ESB provides a fully supported, stable, production ready distribution of the industry leading open source integration framework Apache Camel. Apache Camel uses the well known Enterprise Integration Patterns and Components to make message based system integration simpler yet powerful and scalable.

The Apache Camel uses a lightweight, component based architecture which allows great flexibility in deployment scenarios, for example, stand-alone JVM applications or embedded in a servlet container such as Tomcat, or within a JEE server, or in an OSGi container such as Equinox.

Apache Camel and Talend ESB come out of the box with an impressive set of available components for all commonly used protocols like http, https, ftp, xmpp, rss and many more. A large number of data formats like EDI, JSON, CSV, HL7 and languages like JS, Python, Scala, are supported out of the box. Its extensible architecture allows developers to easily add support for proprietary protocols and data formats.

The Talend ESB distribution supplements Apache Camel with support for OSGi deployment, support for integrating Talend Routes on Camel routes and a number of advanced examples. Its OSGi container uses Apache Karaf, a lightweight container providing advanced features such as provisioning, hot deployment, logger system, dynamic configuration, complete shell environment, and other features.

Talend ESB - Apache Camel - Domain Specific Languages (DSL)

Talend ESB supports the graphical modeling of Routes in Talend Studio. Additionally Talend ESB also supports the deployment of Routes directly written in one of the three Apache Camel DSL:

- Java DSL - A Java based DSL using the fluent builder style. See http://camel.apache.org/java-dsl.html for more information.
- Spring XML - A XML based DSL in Spring XML files. See http://camel.apache.org/spring.html for more information.
- Blueprint XML - A XML based DSL in OSGi Blueprint XML files. See http://camel.apache.org/using-osgi-blueprint-with-camel.html for more information.

Although there are other DSL which is supported by Camel (http://camel.apache.org/dsl.html) and which also works technically with the Talend ESB, it is important to note that Talend ESB only provides full support for the Java, Spring and Blueprint DSL on Talend Runtime and for others only limited support (Minor / S4 Support Level only).

For more information on designing Routes in Talend Studio, see Getting started with a basic Route on Talend Help Center (https://help.talend.com).

Apache Camel Enterprise Integration Patterns

This document lists the Apache Camel Enterprise Integration Patterns (EIPs) supported by Talend ESB.

This release of Talend ESB supports Apache Camel 2.17.6.

Click the links in the table to go to the Apache Camel's website for the latest information of the these EIPs.

Enterprise Integration Patterns: Messaging Systems

Enterprise Integration Pattern	Description
Message Channel (http://camel.apache.org/message-channel.html)	Enables one application to communicate with another using messaging.
Message (http://camel.apache.org/message.html)	Enables two applications connected by a message channel to exchange information.
Pipes and Filters (http://camel.apache.org/pipes-and-filters.html)	Performs complex processing on a message while maintaining independence and flexibility.
Message Router (http://camel.apache.org/message-router.html)	Decouples individual processing steps so that messages can be passed to different filters depending on a set of conditions.
Message Translator (http://camel.apache.org/message-translator.html)	Enables systems using different data formats to communicate with each other using messaging.
Message Endpoint (http://camel.apache.org/message-endpoint.html)	Enables an application connect to a messaging channel to send and receive messages.

Enterprise Integration Patterns: Messaging Channels

Enterprise Integration Pattern	Description
Point to Point Channel (http://camel.apache.org/point-to-point-channel.html)	Enables the caller to be sure that exactly one receiver will receive the document or perform the call.
Publish Subscribe Channel (http://camel.apache.org/publish-subscribe-channel.html)	Enables the sender to broadcast an event to all interested receivers.
Dead Letter Channel (http://camel.apache.org/dead-letter-channel.html)	Enables the messaging system to control the behaviors if a message cannot be delivered.
Guaranteed Delivery (http://camel.apache.org/guaranteed-delivery.html)	Enables the sender to make sure that a message will be delivered, even if the messaging system fails.

Enterprise Integration Pattern	Description
Message Bus (http://camel.apache.org/message-bus.html)	The architecture that enables separate applications to work together, but in a de-coupled fashion such that applications can be easily added or removed without affecting the others.

Enterprise Integration Patterns: Message Construction

Enterprise Integration Pattern	Description
Event Message (http://camel.apache.org/event-message.html)	Enables messaging to be used to transmit events from one application to another.
Request Reply (http://camel.apache.org/request-reply.html)	Enables an applicatin to get a response from the receiver when it sends a message.
Correlation Identifier (http://camel.apache.org/correlation-identifier.html)	Enables a requestor that has received a reply to know which request the reply is for.
Return Address (http://camel.apache.org/return-address.html)	Enables a replier to know where to send the reply.

Enterprise Integration Patterns: Message Routing

Enterprise Integration Pattern	Description
Content Based Router (http://camel.apache.org/content-based-router.html)	Handles the situation where the implementation of a single logical function (for example, inventory check) is spread across multiple physical systems.
Message Filter (http://camel.apache.org/message-filter.html)	Enables a component to avoid receiving uninteresting messages.
Dynamic Router (http://camel.apache.org/dynamic-router.html)	Avoids the dependency of the router on all possible destinations while maintaining its efficiency.
Recipient List (http://camel.apache.org/recipient-list.html)	Routes a message to a list of (static or dynamically) specified recipients.
Splitter (http://camel.apache.org/splitter.html)	Enables each of the multiple elements in a message to be processed in a different way.
Aggregator (http://camel.apache.org/aggregator2.html)	Combines the results of individual, but related messages so that they can be processed as a whole.
Resequencer (http://camel.apache.org/resequencer.html)	Gets a stream of related but out-of-sequence messages back into the correct order.

Enterprise Integration Pattern	Description
Composed Message Processor (http://camel.apache.org/composed-message-processor.html)	Maintains the overall message flow when processing a message consisting of multiple elements, each of which may require different processing.
Scatter-Gather (http://camel.apache.org/scatter-gather.html)	Maintains the overall message flow when a message needs to be sent to multiple recipients, each of which may send a reply.
Routing Slip (http://camel.apache.org/routing-slip.html)	Routes a message consecutively through a series of processing steps when the sequence of steps is not known at design-time and may vary for each message.
Throttler (http://camel.apache.org/throttler.html)	Throttles messages to ensure that a specific endpoint does not get overloaded, or an agreed SLA with some external service is not exeeded.
Sampling (http://camel.apache.org/sampling.html)	Samples one message out of many in a given period to avoid downstream route from getting overloaded.
Delayer (http://camel.apache.org/delayer.html)	Delays the sending of a message.
Load Balancer (http://camel.apache.org/load-balancer.html)	Balance the processing load across a number of endpoints.
Multicast (http://camel.apache.org/multicast.html)	Routes a message to a number of endpoints at the same time.
Loop (http://camel.apache.org/loop.html)	Repeats processing a message in a loop.

Enterprise Integration Patterns: Message Transformation

Enterprise Integration Pattern	Description
Content Enricher (http://camel.apache.org/content-enricher.html)	Communicates with another system if the message originator does not have all the required data items available.
Content Filter (http://camel.apache.org/content-filter.html)	Simplifies dealing with a large message, when you are interested only in a few data items.
Claim Check (http://camel.apache.org/claim-check.html)	Reduces the data volume of message sent across the system without sacrificing information content.
Normalizer (http://camel.apache.org/normalizer.html)	Processes messages that are semantically equivalent, but arrive in a different format.
Sort (http://camel.apache.org/sort.html)	Sorts the body of a message.

Enterprise Integration Pattern	Description
Validate (http://camel.apache.org/validate.html)	Validates a message.

Enterprise Integration Patterns: Messaging Endpoints

Enterprise Integration Pattern	Description
Messaging Mapper (http://camel.apache.org/messaging-mapper.html)	Moves data between domain objects and the messaging infrastructure while keeping the two independent of each other.
Event Driven Consumer (http://camel.apache.org/event-driven-consumer.html)	Enables an application to consume messages automatically as they become available.
Polling Consumer (http://camel.apache.org/polling-consumer.html)	Enables an application to consume a message when the application is ready.
Competing Consumers (http://camel.apache.org/competing-consumers.html)	Enables a messaging client to process multiple messages concurrently.
Message Dispatcher (http://camel.apache.org/message-dispatcher.html)	Enables multiple consumers on a single channel to coordinate their message processing.
Selective Consumer (http://camel.apache.org/selective-consumer.html)	Enables a message consumer to select which messages it wishes to receive.
Durable Subscriber (http://camel.apache.org/durable-subscriber.html)	Enables a subscriber to avoid missing messages while it's not listening for them.
Idempotent Consumer (http://camel.apache.org/idempotent-consumer.html)	Enables a message receiver to deal with duplicate messages.
Transactional Client (http://camel.apache.org/transactional-client.html)	Enables a client to control its transactions with the messaging system.
Messaging Gateway (http://camel.apache.org/messaging-gateway.html)	Encapsulates access to the messaging system from the rest of the application.
Service Activator (http://camel.apache.org/service-activator.html)	Enables an application to design a service that can be invoked both via various messaging technologies and via non-messaging techniques.

Enterprise Integration Patterns: System Management

Enterprise Integration Pattern	Description
ControlBus (http://camel.apache.org/controlbus.html)	Effectively administer a messaging system that is distributed across multiple platforms and a wide geographic area.
Detour (http://camel.apache.org/detour.html)	Routes a message through intermediate steps to perform validation, testing or debugging functions.
Wire Tap (http://camel.apache.org/wire-tap.html)	Inspects messages that travel on a point-to-point channel.
Message History (http://camel.apache.org/message-history.html)	Enables analyzing and debugging the flow of messages in a loosely coupled system.
Log (http://camel.apache.org/logeip.html)	Logs the processing a message.

Apache Camel Components

This document lists the Camel components supported by Talend ESB.

This release of Talend ESB supports Apache Camel 2.17.6.

Click the links in the table to go to the Apache Camel's website for the latest information of the these components.

Component / ArtifactId / URI	Description
AHC (camel-ahc) `ahc:httpUri`	To call external HTTP services using Async Http Client.
AHC Websocket (camel-ahc-ws) `ahc-ws:httpUri`	To exchange data with external Websocket servers using Async Http Client.
APNS (camel-apns) `apns:name`	For sending notifications to Apple iOS devices.
Avro (camel-avro) `avro:transport:host:port/messageName`	Working with Apache Avro for data serialization.
Atom (camel-atom) `atom:feedUri`	For consuming Atom RSS feeds.
Atmosphere Websocket (camel-atmosphere-websocket) `atmosphere-websocket:servicePath`	To exchange data with external Websocket clients using Atmosphere.
Braintree (camel-braintree) `braintree:apiName/methodName`	For integrating with the Braintree Payment System.
Camel Context (camel-context) `context:contextId:localEndpointUrl`	To send/receive messages between Camel routes in a black box way. This component is deprecated.
CMIS (camel-cmis) `cmis:cmsUrl`	The cmis component uses the Apache Chemistry client API and allows you to add/read nodes to/from a CMIS compliant content repositories.
CoAP (camel-coap) `coap:uri`	For sending and receiving messages from COAP capable devices.
CouchDB (camel-couchdb) `couchdb:protocol:hostname:port/database`	To integrate with CouchDB databases.
Crypto (JCE) (camel-crypto) `crypto:cryptoOperation:name`	For signing and verifying exchanges using the Signature Service of the Java Cryptographic Extension (JCE).
CXF (camel-cxf) `cxf:beanId:address`	Works for SOAP WebServices using Apache CXF.
Disruptor (camel-disruptor) `disruptor:name`	To provide asynchronous SEDA behavior using LMAX Disruptor.

Component / ArtifactId / URI	Description
EHCache (camel-cache) `cache:cacheName`	To perform caching operations using EHCache as the Cache Implementation. This component is deprecated.
Elasticsearch (camel-elasticsearch) `elasticsearch:clusterName`	For interfacing with ElasticSearch server.
ElSQL (camel-elsql) `elsql:elsqlName:resourceUri`	The ElSQL component is an extension to the existing SQL Component that uses ElSql to define the SQL queries.
etcd (camel-etcd) `etcd:namespace/path`	To work with Etcd, a distributed reliable key-value store.
Exec (camel-exec) `exec:executable`	To execute OS system commands.
Facebook (camel-facebook) `facebook:methodName`	To provide access to all of the Facebook APIs accessible using Facebook4J.
Flatpack (camel-flatpack) `flatpack:type:resourceUri`	The flatpack component supports fixed width and delimited file parsing via the FlatPack library.
FOP (camel-fop) `fop:outputType`	To render a message into different output formats using Apache FOP.
Freemarker (camel-freemarker) `freemarker:resourceUri`	Transforms the message using a FreeMarker template.
FTP (camel-ftp) `ftp:host:port/directoryName`	For uploading or downloading files from FTP servers.
Geocoder (camel-geocoder) `geocoder:address:latlng`	For looking up geocodes (latitude and longitude) for a given address or reverse lookup.
Git (camel-git) `git:localPath`	For working with git repositories.
Guava EventBus (camel-guava-eventbus) `guava-eventbus:eventBusRef`	Provides integration bridge between Camel and Google Guava EventBus.
Grape (camel-grape) `grape:defaultCoordinates`	To fetch, load and manage additional jars when CamelContext is running.
HBase (camel-hbase) `hbase:tableName`	For reading/writing from/to an HBase store (Hadoop database).
HDFS (camel-hdfs) `hdfs:hostName:port/path`	For reading/writing from/to an HDFS file system using Hadoop 1.x. This component is deprecated.
HDFS2 (camel-hdfs2) `hdfs2:hostName:port/path`	For reading/writing from/to an HDFS file system using Hadoop 2.x.

Component / ArtifactId / URI	Description
HTTP4 (camel-http4) `http4:httpUri`	For calling out to external HTTP servers using Apache HTTP Client 4.x.
Ignite Cache (camel-ignite) `ignite-cache:cacheName`	The Ignite Cache endpoint is one of camel-ignite endpoints which allows you to interact with an Ignite Cache.
Infinispan (camel-infinispan) `infinispan:cacheName`	For reading/writing from/to Infinispan distributed key/value store and data grid.
IronMQ (camel-ironmq) `ironmq:queueName`	The ironmq provides integration with IronMQ an elastic and durable hosted message queue as a service.
JBPM (camel-jbpm) `jbpm:connectionURL`	Provides integration with jBPM (Business Process Management).
JCache (camel-jcache) `jcache:cacheName`	To perform caching operations using JSR107/JCache as cache implementation.
JCR (camel-jcr) `jcr:host/base`	To add/read nodes to/from a JCR compliant content repository.
JDBC (camel-jdbc) `jdbc:dataSourceName`	To access databases through JDBC where SQL queries are sent in the message body.
Jetty 9 (camel-jetty9) `jetty:httpUri`	Provides HTTP-based endpoints for consuming and producing HTTP requests.
JGroups (camel-jgroups) `jgroups:clusterName`	Provides exchange of messages between Camel and JGroups clusters.
JMS (camel-jms) `jms:destinationType:destinationName`	Allows messages to be sent to, or consumed from a JMS Queue or Topic.
JMX (camel-jmx) `jmx:serverURL`	To receive JMX notifications.
JOLT (camel-jolt) `jolt:resourceUri`	To process JSON messages using an JOLT specification (such as JSON-JSON transformation).
JPA (camel-jpa) `jpa:entityType`	To store and retrieve Java objects from databases using JPA.
Jsch (camel-jsch) `scp:host:port/directoryName`	To copy files using the secure copy protocol (SCP).
Kafka (camel-kafka) `kafka:topic`	Allows messages to be sent to, or consumed from Apache Kafka brokers.
Krati (camel-krati) `krati:path`	Allows the use of krati datastores and datasets inside Camel.

Component / ArtifactId / URI	Description
Kubernetes (camel-kubernetes) `kubernetes:masterUrl`	To work with Kubernetes PaaS.
Lucene (camel-lucene) `lucene:host:operation`	To insert or query from Apache Lucene databases.
Mail (camel-mail) `imap:host:port`	To send or receive emails using imap/pop3 or smtp protocols.
Mina2 (camel-mina2) `mina2:protocol:host:port`	Socket level networking using TCP or UDP with the Apache Mina 2.x library.
MLLP (camel-mllp) `mllp:hostname:port`	Provides functionality required by Healthcare providers to communicate with other systems using the MLLP protocol.
Mock (camel-core) `mock:name`	For testing routes and mediation rules using mocks.
MongoDB (camel-mongodb) `mongodb:connectionBean`	For working with documents stored in MongoDB database.
MongoDB GridFS (camel-mongodb-gridfs) `mongodb-gridfs:connectionBean`	For working with MongoDB GridFS.
MQTT (camel-mqtt) `mqtt:name`	For communicating with MQTT M2M message brokers using FuseSource MQTT Client.
Mustache (camel-mustache) `mustache:resourceUri`	Transforms the message using a Mustache template.
MyBatis (camel-mybatis) `mybatis:statement`	Performs a query, poll, insert, update or delete in a relational database using MyBatis.
Nats (camel-nats) `nats:servers`	Produces and consumes messages from NATS.
Netty (camel-netty) `netty:protocol:host:port`	Socket level networking using TCP or UDP with the Netty 3.x library. This component is deprecated.
OpenShift (camel-openshift) `openshift:clientId`	To Manage your Openshift 2.x applications. This component is deprecated.
OptaPlanner (camel-optaplanner) `optaplanner:configFile`	Solves the planning problem contained in a message with OptaPlanner.
Paho (camel-paho) `paho:topic`	For communicating with MQTT M2M message brokers using Eclipse Paho MQTT Client.
PDF (camel-pdf) `pdf:operation`	Provides the ability to create, modify or extract content from PDF documents.

Component / ArtifactId / URI	Description
Quartz (camel-quartz) `quartz:groupName/timerName`	Provides a scheduled delivery of messages using the Quartz 1.x scheduler. This component is deprecated.
Quartz2 (camel-quartz2) `quartz2:groupName/triggerName`	Provides a scheduled delivery of messages using the Quartz 2.x scheduler.
RabbitMQ (camel-rabbitmq) `rabbitmq:hostname:portNumber/exchangeName`	To produce and consume messages from RabbitMQ instances.
RMI (camel-rmi) `rmi:hostname:port/name`	For invoking Java RMI beans from Camel.
RSS (camel-rss) `rss:feedUri`	For consuming RSS feeds.
Salesforce (camel-salesforce) `salesforce:operationName:topicName`	For integrating Camel with the massive Salesforce API.
SAP NetWeaver (camel-sap-netweaver) `sap-netweaver:url`	To integrate with the SAP NetWeaver Gateway using HTTP transports.
Servlet (camel-servlet) `servlet:contextPath`	To use a HTTP Servlet as entry for Camel routes when running in a servlet container.
Simple JMS (camel-sjms) `sjms:destinationType:destinationName`	Allows messages to be sent to, or consumed from a JMS Queue or Topic (uses JMS 1.x API).
Slack (camel-slack) `slack:channel`	To send messages to Slack.
SMPP (camel-smpp) `smpp:host:port`	To send and receive SMS using a SMSC (Short Message Service Center).
SNMP (camel-snmp) `snmp:host:port`	To poll SNMP capable devices or receiving traps.
Solr (camel-solr) `solr:url`	To interface with an Apache Lucene Solr server.
Splunk (camel-splunk) `splunk:name`	To publish or search for events in Splunk.
Spring Batch (camel-spring-batch) `spring-batch:jobName`	To send messages to Spring Batch for further processing.
Spring Integration (camel-spring-integration) `spring-integration:defaultChannel`	Bridges Camel with Spring Integration.
Spring LDAP (camel-spring-ldap) `spring-ldap:templateName`	To perform searches in LDAP servers using filters as the message payload.

Component / ArtifactId / URI	Description
Spring Redis (camel-spring-redis) `spring-redis:host:port`	To send and receive messages from Redis.
Spring WebService (camel-spring-ws) `spring-ws:type:lookupKey:webServiceEndpointUri`	Works for SOAP WebServices using Spring WebServices.
SQL (camel-sql) `sql:query`	To work with databases using JDBC SQL queries.
SSH (camel-ssh) `ssh:host:port`	The ssh component enables access to SSH servers so that you can send an SSH command and process the response.
StAX (camel-stax) `stax:contentHandlerClass`	Allows messages to be process through a SAX ContentHandler.
Stomp (camel-stomp) `stomp:destination`	For communicating with Stomp compliant message brokers.
Twitter (camel-twitter) `twitter:kind`	Integrates with Twitter to send tweets or search for tweets and more.
Undertow (camel-undertow) `undertow:httpURI`	Provides HTTP-based endpoints for consuming and producing HTTP requests.
Velocity (camel-velocity) `velocity:resourceUri`	Transforms the message using a Velocity template.
Vertx (camel-vertx) `vertx:address`	For sending and receive messages from a vertx event bus.
Weather (camel-weather) `weather:name`	Polls the weather information from Open Weather Map.
Yammer (camel-yammer) `yammer:function`	To interact with the Yammer enterprise social network.
ZooKeeper (camel-zookeeper) `zookeeper:serverUrls/path`	Allows interaction with a ZooKeeper cluster.

Talend ESB Mediation Examples

This document lists the mediation examples provided by Talend ESB.

The samples folder of the Talend ESB download contains examples that are provided by the Apache Camel project, as well as Talend ESB-specific examples showing multiple usages of Camel routing. Each Talend ESB sample has its own README file providing a full description of the sample along with deployment information using embedded Jetty or Talend Runtime Container. The examples provided by the Apache Camel project and bundled with the Talend ESB are listed and explained on the Camel website (http://camel.apache.org/examples.html). The following table provides a summary of additional mediation examples provided in the Talend ESB distribution.

Example	Description
blueprint	Provides an example of deploying Camel routes as an OSGi bundle in the container.
claimcheck	EAI patterns example demonstrating use of the Claim Check, Splitter, Reswquencer and Delayer patterns.
jaxrs-jms-http	Shows how a JAX-RS service can be offered an used with Camel transports.
jaxws-jms	Shows how to publish and call a CXF service using SOAP/ JMS using Camel as a CXF transport.
spring-security	Example shows how to leverage Spring Security to secure Camel routes in general and also specifically when combined with CXF JAX-WS and JAX-RS endpoints.