### Components Included

Camel includes the following [Component](http://camel.apache.org/component.html) implementations via [URIs](http://camel.apache.org/uris.html).

important

Make sure to read [How do I configure endpoints](http://camel.apache.org/how-do-i-configure-endpoints.html) to learn more about configuring endpoints. For example how to refer to beans in the [Registry](http://camel.apache.org/registry.html) or how to use raw values for password options, and using [property placeholders](http://camel.apache.org/using-propertyplaceholder.html) etc.

|  |  |  |
| --- | --- | --- |
| **Component / ArtifactId / URI** | | **Description** |
| [AHC](http://camel.apache.org/ahc.html) / camel-ahc   |  | | --- | | ahc:http[s]://hostName[:port][/resourceUri][?options] | | | To call external HTTP services using [Async Http Client](http://github.com/sonatype/async-http-client) |
| [AHC-WS](http://camel.apache.org/ahc-ws.html) / camel-ahc-ws   |  | | --- | | ahc-ws[s]://hostName[:port][/resourceUri][?options] | | | To exchange data with external Websocket servers using [Async Http Client](http://github.com/sonatype/async-http-client) |
| [AMQP](http://camel.apache.org/amqp.html) / camel-amqp   |  | | --- | | amqp:[queue:|topic:]destinationName[?options] | | | For Messaging with [AMQP protocol](http://www.amqp.org/) |
| [APNS](http://camel.apache.org/apns.html) / camel-apns   |  | | --- | | apns:<notify|consumer>[?options] | | | For sending notifications to Apple iOS devices |
| [Atmosphere-Websocket](http://camel.apache.org/atmosphere-websocket.html)  / camel-atmosphere-websocket   |  | | --- | | atmosphere-[websocket:///relative](websocket://relative) path[?options] | | | To exchange data with external Websocket clients using [Atmosphere](https://github.com/Atmosphere/atmosphere) |
| [Atom](http://camel.apache.org/atom.html) / camel-atom   |  | | --- | | atom:atomUri[?options] | | | Working with [Apache Abdera](http://incubator.apache.org/abdera/) for atom integration, such as consuming an atom feed. |
| [Avro](http://camel.apache.org/avro.html) / camel-avro   |  | | --- | | avro:[transport]:[host]:[port][/messageName][?options] | | | Working with [Apache Avro](http://avro.apache.org/) for data serialization. |
| [AWS-CW](http://camel.apache.org/aws-cw.html) / [camel-aws](http://camel.apache.org/aws.html)   |  | | --- | | aws-<cw://namespace>[?options] | | | For working with [Amazon's CloudWatch (CW)](http://aws.amazon.com/cloudwatch/). |
| [AWS-DDB](http://camel.apache.org/aws-ddb.html) / [camel-aws](http://camel.apache.org/aws.html)   |  | | --- | | aws-<ddb://tableName>[?options] | | | For working with [Amazon's DynamoDB (DDB)](http://aws.amazon.com/dynamodb/). |
| [AWS-EC2](http://camel.apache.org/aws-ec2.html) / [camel-aws](http://camel.apache.org/aws.html)   |  | | --- | | aws-<ec2://label>[?options] | | | For working with [Amazon's Elastic Compute Cloud (EC2)](http://aws.amazon.com/ec2/). |
| [AWS-SDB](http://camel.apache.org/aws-sdb.html) / [camel-aws](http://camel.apache.org/aws.html)   |  | | --- | | aws-<sdb://domainName>[?options] | | | For working with [Amazon's SimpleDB (SDB)](http://aws.amazon.com/simpledb/). |
| [AWS-SES](http://camel.apache.org/aws-ses.html) / [camel-aws](http://camel.apache.org/aws.html)   |  | | --- | | aws-<ses://from>[?options] | | | For working with [Amazon's Simple Email Service (SES)](http://aws.amazon.com/ses/). |
| [AWS-SNS](http://camel.apache.org/aws-sns.html) / [camel-aws](http://camel.apache.org/aws.html)   |  | | --- | | aws-<sns://topicName>[?options] | | | For Messaging with [Amazon's Simple Notification Service (SNS)](http://aws.amazon.com/sns/). |
| [AWS-SQS](http://camel.apache.org/aws-sqs.html) / [camel-aws](http://camel.apache.org/aws.html)   |  | | --- | | aws-<sqs://queueName>[?options] | | | For Messaging with [Amazon's Simple Queue Service (SQS)](http://aws.amazon.com/sqs/). |
| [AWS-SWF](http://camel.apache.org/aws-swf.html) / [camel-aws](http://camel.apache.org/aws.html)   |  | | --- | | aws-[swf://](NULL)<worfklow|activity>[?options] | | | For Messaging with [Amazon's Simple Workflow Service (SWF)](http://aws.amazon.com/swf/). |
| [AWS-S3](http://camel.apache.org/aws-s3.html) / [camel-aws](http://camel.apache.org/aws.html)   |  | | --- | | aws-<s3://bucketName>[?options] | | | For working with [Amazon's Simple Storage Service (S3)](http://aws.amazon.com/s3/). |
| [Bean](http://camel.apache.org/bean.html) / camel-core   |  | | --- | | bean:beanName[?options] | | | Uses the [Bean Binding](http://camel.apache.org/bean-binding.html) to bind message exchanges to beans in the [Registry](http://camel.apache.org/registry.html). Is also used for exposing and invoking POJO (Plain Old Java Objects). |
| [Beanstalk](http://camel.apache.org/beanstalk.html) / camel-beanstalk   |  | | --- | | beanstalk:hostname:port/tube[?options] | | | For working with [Amazon's Beanstalk](http://aws.amazon.com/elasticbeanstalk/). |
| [Bean Validator](http://camel.apache.org/bean-validator.html) / camel-bean-validator   |  | | --- | | bean-validator:label[?options] | | | Validates the payload of a message using the Java Validation API ([JSR 303](http://jcp.org/en/jsr/detail?id=303) and JAXP Validation) and its reference implementation [Hibernate Validator](http://docs.jboss.org/hibernate/stable/validator/reference/en/html_single/) |
| [Box](http://camel.apache.org/box.html) / camel-box   |  | | --- | | <box://endpoint-prefix/endpoint?>[options] | | | For uploading, downloading and managing files, managing files, folders, groups, collaborations, etc. on Box.com. |
| [Browse](http://camel.apache.org/browse.html) / camel-core   |  | | --- | | browse:someName | | | Provides a simple [BrowsableEndpoint](http://camel.apache.org/browsableendpoint.html) which can be useful for testing, visualisation tools or debugging. The exchanges sent to the endpoint are all available to be browsed. |
| [Cache](http://camel.apache.org/cache.html) / camel-cache   |  | | --- | | <cache://cacheName>[?options] | | | The cache component facilitates creation of caching endpoints and processors using [EHCache](http://ehcache.org/) as the cache implementation. |
| [Cassandra](http://camel.apache.org/cassandra.html) / camel-cassandraql   |  | | --- | | cql:localhost/keyspace | | | For integrating with [Apache Cassandra](http://cassandra.apache.org/). |
| [Class](http://camel.apache.org/class.html) / camel-core   |  | | --- | | class:className[?options] | | | Uses the [Bean Binding](http://camel.apache.org/bean-binding.html) to bind message exchanges to beans in the [Registry](http://camel.apache.org/registry.html). Is also used for exposing and invoking POJO (Plain Old Java Objects). |
| [Chunk](http://camel.apache.org/chunk.html) / camel-chunk   |  | | --- | | chunk:templateName[?options] | | | Generates a response using a [Chunk](http://www.x5software.com/chunk/examples/ChunkExample) template |
| [CMIS](http://camel.apache.org/cmis.html) / camel-cmis   |  | | --- | | <cmis://cmisServerUrl>[?options] | | | Uses the [Apache Chemistry](http://chemistry.apache.org/java/opencmis.html) client API to interface with CMIS supporting CMS |
| [Cometd](http://camel.apache.org/cometd.html) / camel-cometd   |  | | --- | | <cometd://hostName:port/channelName>[?options] | | | Used to deliver messages using the [jetty cometd implementation](http://docs.codehaus.org/display/JETTY/Cometd+%28aka+Bayeux%29) of the [bayeux protocol](http://svn.xantus.org/shortbus/trunk/bayeux/bayeux.html) |
| [Context](http://camel.apache.org/context.html) / camel-context   |  | | --- | | context:camelContextId:localEndpointName[?options] | | | Used to refer to endpoints within a separate CamelContext to provide a simple [black box composition](http://camel.apache.org/context.html) approach so that routes can be combined into a CamelContext and then used as a black box component inside other routes in other CamelContexts |
| [ControlBus](http://camel.apache.org/controlbus-component.html) / camel-core   |  | | --- | | controlbus:command[?options] | | | [ControlBus](http://camel.apache.org/controlbus.html) EIP that allows to send messages to [Endpoint](http://camel.apache.org/endpoint.html)s for managing and monitoring your Camel applications. |
| [CouchDB](http://camel.apache.org/couchdb.html) / camel-couchdb   |  | | --- | | couchdb:hostName[:port]/database[?options] | | | To integrate with [Apache CouchDB](http://couchdb.apache.org/). |
| [Crypto (Digital Signatures)](http://camel.apache.org/crypto-digital-signatures.html) / camel-crypto   |  | | --- | | crypto:<sign|verify>:name[?options] | | | Used to sign and verify exchanges using the Signature Service of the Java Cryptographic Extension. |
| [CXF](http://camel.apache.org/cxf.html) / camel-cxf   |  | | --- | | cxf:<bean:cxfEndpoint|//someAddress>[?options] | | | Working with [Apache CXF](http://apache.org/cxf/) for web services integration |
| [CXF Bean](http://camel.apache.org/cxf-bean-component.html) / camel-cxf   |  | | --- | | cxfbean:serviceBeanRef[?options] | | | Proceess the exchange using a JAX WS or JAX RS annotated bean from the registry. Requires less configuration than the above CXF Component |
| [CXFRS](http://camel.apache.org/cxfrs.html) / camel-cxf   |  | | --- | | cxfrs:<bean:rsEndpoint|//address>[?options] | | | Working with [Apache CXF](http://apache.org/cxf/) for REST services integration |
| [DataFormat](http://camel.apache.org/dataformat-component.html) / camel-core   |  | | --- | | dataformat:name:<marshal|unmarshal>[?options] | | | for working with [Data Format](http://camel.apache.org/data-format.html)s as if it was a regular Component supporting Endpoints and URIs. |
| [DataSet](http://camel.apache.org/dataset.html) / camel-core   |  | | --- | | dataset:name[?options] | | | For load & soak testing the [DataSet](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/component/dataset/DataSet.html) provides a way to create huge numbers of messages for sending to [Components](http://camel.apache.org/components.html) or asserting that they are consumed correctly |
| [Direct](http://camel.apache.org/direct.html) / camel-core   |  | | --- | | direct:someName[?options] | | | Synchronous call to another endpoint from **same** CamelContext. |
| [Direct-VM](http://camel.apache.org/direct-vm.html) / camel-core   |  | | --- | | direct-vm:someName[?options] | | | Synchronous call to another endpoint in another CamelContext running in the same JVM. |
| [DNS](http://camel.apache.org/dns.html) / camel-dns   |  | | --- | | dns:operation[?options] | | | To lookup domain information and run DNS queries using [DNSJava](http://www.xbill.org/dnsjava/) |
| [Disruptor](http://camel.apache.org/disruptor.html) / camel-disruptor   |  | | --- | | disruptor:someName[?<option>]  disruptor-vm:someName[?<option>] | | | To provide the implementation of [SEDA](http://camel.apache.org/seda.html) which is based on [disruptor](https://github.com/LMAX-Exchange/disruptor) |
| [Docker](http://camel.apache.org/docker.html) / camel-docker   |  | | --- | | docker://[operation]?[options] | | | To communicate with [Docker](https://www.docker.com/) |
| [Dozer](http://camel.apache.org/dozer.html) / camel-dozer   |  | | --- | | dozer://name?[options] | | | To convert message body using the Dozer type converter library. |
| [D](http://camel.apache.org/disruptor.html)[ropbox](http://camel.apache.org/dropbox.html) / camel-dropbox   |  | | --- | | dropbox://[operation]?[options] | | | The **dropbox:** component allows you to treat [Dropbox](https://www.dropbox.com/) remote folders as a producer or consumer of messages. |
| [EJB](http://camel.apache.org/ejb.html) / camel-ejb   |  | | --- | | ejb:ejbName[?options] | | | Uses the [Bean Binding](http://camel.apache.org/bean-binding.html) to bind message exchanges to EJBs. It works like the [Bean](http://camel.apache.org/bean.html) component but just for accessing EJBs. Supports EJB 3.0 onwards. |
| [ElasticSearch](http://camel.apache.org/elasticsearch.html) / camel-elasticsearch   |  | | --- | | <elasticsearch://clusterName>[?options] | | | For interfacing with an [ElasticSearch](http://elasticsearch.org) server. |
| [Spring Event](http://camel.apache.org/spring-event.html) / camel-spring   |  | | --- | | spring-<event://default> | | | Working with Spring ApplicationEvents |
| [EventAdmin](http://camel.apache.org/eventadmin.html) / camel-eventadmin   |  | | --- | | eventadmin:topic[?options] | | | Receiving OSGi EventAdmin events |
| [Exec](http://camel.apache.org/exec.html) / camel-exec   |  | | --- | | <exec://executable>[?options] | | | For executing system commands |
| [Facebook](http://camel.apache.org/facebook.html) / camel-facebook   |  | | --- | | <facebook://endpoint>[?options] | | | Providing access to all of the Facebook APIs accessible using [Facebook4J](http://facebook4j.org/en/index.html) |
| [File](http://camel.apache.org/file2.html) / camel-core   |  | | --- | | [file://nameOfFileOrDirectory](file:///\\nameOfFileOrDirectory)[?options] | | | Sending messages to a file or polling a file or directory. |
| [Flatpack](http://camel.apache.org/flatpack.html) / camel-flatpack   |  | | --- | | flatpack:[fixed|delim]:configFile[?options] | | | Processing fixed width or delimited files or messages using the [FlatPack library](http://flatpack.sourceforge.net) |
| [FOP](http://camel.apache.org/fop.html) / camel-fop   |  | | --- | | fop:outputFormat[?options] | | | Renders the message into different output formats using [Apache FOP](http://xmlgraphics.apache.org/fop/index.html) |
| [FreeMarker](http://camel.apache.org/freemarker.html) / camel-freemarker   |  | | --- | | freemarker:templateName[?options] | | | Generates a response using a [FreeMarker](http://freemarker.org/) template |
| [FTP](http://camel.apache.org/ftp2.html) / camel-ftp   |  | | --- | | ftp:contextPath[?options] | | | Sending and receiving files over FTP. |
| [FTPS](http://camel.apache.org/ftp2.html) / camel-ftp   |  | | --- | | [ftps://](NULL)[username@]hostName[:port]/directoryName[?options] | | | Sending and receiving files over FTP Secure (TLS and SSL). |
| [Ganglia](http://camel.apache.org/ganglia.html) / camel-ganglia  ganglia:destination:port[?options] | | Sends values as metrics to the [Ganglia](http://ganglia.info) performance monitoring system using [gmetric4j](https://github.com/ganglia/gmetric4j).  Can be used along with [JMXetric](https://github.com/ganglia/jmxetric). |
| [GAuth](http://camel.apache.org/gauth.html) / [camel-gae](http://camel.apache.org/gae.html)   |  | | --- | | <gauth://name>[?options] | | | Used by web applications to implement an [OAuth](http://code.google.com/apis/accounts/docs/OAuth.html) consumer. See also [Camel Components for Google App Engine](http://camel.apache.org/gae.html). |
| [GHttp](http://camel.apache.org/ghttp.html) / [camel-gae](http://camel.apache.org/gae.html)   |  | | --- | | ghttp:contextPath[?options] | | | Provides connectivity to the [URL fetch service](http://code.google.com/appengine/docs/java/urlfetch/) of Google App Engine but can also be used to receive messages from servlets. See also [Camel Components for Google App Engine](http://camel.apache.org/gae.html). |
| [Git](http://camel.apache.org/git.html) / [camel-git](http://camel.apache.org/git.html)   |  | | --- | | git:localRepositoryPath[?options] | | | Supports interaction with [Git](https://git-scm.com/) repositories |
| [Github](http://camel.apache.org/github.html) / [camel-github](http://camel.apache.org/github.html)   |  | | --- | | github:endpoint[?options] | | | Supports interaction with [Github](https://github.com/) |
| [GLogin](http://camel.apache.org/glogin.html) / [camel-gae](http://camel.apache.org/gae.html)   |  | | --- | | <glogin://hostname>[:port][?options] | | | Used by Camel applications outside Google App Engine (GAE) for programmatic login to GAE applications. See also [Camel Components for Google App Engine](http://camel.apache.org/gae.html). |
| [GTask](http://camel.apache.org/gtask.html) / [camel-gae](http://camel.apache.org/gae.html)   |  | | --- | | <gtask://queue-name>[?options] | | | Supports asynchronous message processing on Google App Engine by using the [task queueing service](http://code.google.com/appengine/docs/java/taskqueue/) as message queue. See also [Camel Components for Google App Engine](http://camel.apache.org/gae.html). |
| [Google Calendar](http://camel.apache.org/googlecalendar.html) / [camel-google-calendar](http://camel.apache.org/googlecalendar.html)   |  | | --- | | google-calendar://endpoint-prefix/endpoint?[options] | | | Supports interaction with [Google Calendar's REST API](https://developers.google.com/google-apps/calendar/v3/reference/). |
| [Google Drive](http://camel.apache.org/googledrive.html) / [camel-google-drive](http://camel.apache.org/googledrive.html)   |  | | --- | | google-drive://endpoint-prefix/endpoint?[options] | | | Supports interaction with [Google Drive's REST API](https://developers.google.com/drive/v2/reference/). |
| [Google Mail](http://camel.apache.org/googlemail.html) / [camel-google-mail](http://camel.apache.org/googlemail.html)   |  | | --- | | google-<mail://endpoint-prefix/endpoint?>[options] | | | Supports interaction with [Google Mail's REST API](https://developers.google.com/gmail/api/v1/reference/). |
| [GMail](http://camel.apache.org/gmail.html) / [camel-gae](http://camel.apache.org/gae.html)   |  | | --- | | <gmail://user@g>[oogle]mail.com[?options] | | | Supports sending of emails via the [mail service](http://code.google.com/appengine/docs/java/mail/) of Google App Engine. See also [Camel Components for Google App Engine](http://camel.apache.org/gae.html). |
| [Gora](http://camel.apache.org/gora.html)/ camel-gora   |  | | --- | | gora:instanceName[?options] | | | Supports to work with NoSQL databases using the [Apache Gora](http://gora.apache.org/) framework. |
| [G](http://camel.apache.org/gora.html)[rape](http://camel.apache.org/grape.html)/ camel-grape   |  | | --- | | grape:defaultMavenCoordinates | | | [Grape](http://docs.groovy-lang.org/latest/html/documentation/grape.html) component allows you to fetch, load and manage additional jars when CamelContext is running. |
| [Geocoder](http://camel.apache.org/geocoder.html) / camel-geocoder   |  | | --- | | geocoder:<address|latlng:latitude,longitude>[?options] | | | Supports looking up geocoders for an address, or reverse lookup geocoders from an address. |
| [GitHub](http://camel.apache.org/github.html) / camel-github   |  | | --- | | <github://endpoint>[?options] | | | For interacting with GitHub |
| [Google Guava EventBus](http://camel.apache.org/guava-eventbus.html) / camel-guava-eventbus   |  | | --- | | guava-eventbus:busName[?options] | | | The [Google Guava EventBus](http://docs.guava-libraries.googlecode.com/git/javadoc/com/google/common/eventbus/package-summary.html) allows publish-subscribe-style communication between components without requiring the components to explicitly register with one another (and thus be aware of each other). This component provides integration bridge between Camel and [Google Guava EventBus](http://docs.guava-libraries.googlecode.com/git/javadoc/com/google/common/eventbus/package-summary.html) infrastructure. |
| [Hazelcast](http://camel.apache.org/hazelcast-component.html) / [camel-hazelcast](http://camel.apache.org/hazelcast-component.html)   |  | | --- | | [hazelcast://](NULL)[type]:cachename[?options] | | | [Hazelcast](http://www.hazelcast.com) is a data grid entirely implemented in Java (single jar). This component supports map, multimap, seda, queue, set, atomic number and simple cluster support. |
| [HBase](http://camel.apache.org/hbase.html) / camel-hbase   |  | | --- | | <hbase://table>[?options] | | | For reading/writing from/to an [HBase](http://hadoop.apache.org/hbase/) store (Hadoop database) |
| [HDFS](http://camel.apache.org/hdfs.html) / camel-hdfs   |  | | --- | | <hdfs://hostName>[:port][/path][?options] | | | For reading/writing from/to an [HDFS](http://hadoop.apache.org/hdfs/) filesystem using Hadoop 1.x |
| [HDFS2](http://camel.apache.org/hdfs2.html) / camel-hdfs2   |  | | --- | | <hdfs2://hostName>[:port][/path][?options] | | | For reading/writing from/to an [HDFS](http://hadoop.apache.org/hdfs/) filesystem using Hadoop 2.x |
| [Hipchat](http://camel.apache.org/hipchat.html) / camel-hipchat   |  | | --- | | [hipchat://](NULL)[host][:port]?options | | | For sending/receiving messages to [Hipchat](https://www.hipchat.com) using v2 API |
| [HL7](http://camel.apache.org/hl7.html) / camel-hl7   |  | | --- | | mina2:<tcp://hostName>[:port][?options] | | | For working with the HL7 MLLP protocol and the HL7 model using the [HAPI library](http://hl7api.sourceforge.net) |
| [Infinispan](http://camel.apache.org/infinispan.html) / camel-infinispan   |  | | --- | | <infinispan://hostName>[?options] | | | For reading/writing from/to [Infinispan](http://infinispan.org/) distributed key/value store and data grid |
| [HTTP](http://camel.apache.org/http.html) / camel-http   |  | | --- | | http:hostName[:port][/resourceUri][?options] | | | For calling out to external HTTP servers using Apache HTTP Client 3.x |
| [HTTP4](http://camel.apache.org/http4.html) / camel-http4   |  | | --- | | http4:hostName[:port][/resourceUri][?options] | | | For calling out to external HTTP servers using Apache HTTP Client 4.x |
| [iBATIS](http://camel.apache.org/ibatis.html) / camel-ibatis   |  | | --- | | <ibatis://statementName>[?options] | | | Performs a query, poll, insert, update or delete in a relational database using [Apache iBATIS](http://ibatis.apache.org/) |
| [IMAP](http://camel.apache.org/mail.html) / camel-mail   |  | | --- | | [imap://](NULL)[username@]hostName[:port][?options] | | | Receiving email using [IMAP](http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol) |
| [IMAPS](http://camel.apache.org/mail.html) / camel-mail   |  | | --- | | [imaps://](NULL)[username@]hostName[:port][?options] | | | ... |
| [IRC](http://camel.apache.org/irc.html) / camel-irc   |  | | --- | | irc:[login@]hostName[:port]/#room[?options] | | | For IRC communication |
| [JavaSpace](http://camel.apache.org/javaspace.html) / camel-javaspace   |  | | --- | | javaspace:<jini://hostName>[?options] | | | Sending and receiving messages through [JavaSpace](http://java.sun.com/products/jini/2.1/doc/specs/html/js-spec.html) |
| [jBPM](http://camel.apache.org/javaspace.html) / camel-jbpm   |  | | --- | | jbpm:hostName[:port][/resourceUri][?options] | | | Sending messages through kie-remote-client API to jBPM. |
| [jclouds](http://camel.apache.org/jclouds.html) / camel-jclouds   |  | | --- | | jclouds:<blobstore|compute>:[provider id][?options] | | | For interacting with cloud compute & blobstore service via [jclouds](http://www.jclouds.org) |
| [JCR](http://camel.apache.org/jcr.html) / camel-jcr   |  | | --- | | <jcr://user:password@repository/path/to/node>[?options] | | | Storing a message in a JCR compliant repository like [Apache Jackrabbit](http://jackrabbit.apache.org) |
| [JDBC](http://camel.apache.org/jdbc.html) / camel-jdbc   |  | | --- | | jdbc:dataSourceName[?options] | | | For performing JDBC queries and operations |
| [Jetty](http://camel.apache.org/jetty.html) / camel-jetty   |  | | --- | | jetty:hostName[:port][/resourceUri][?options] | | | For exposing services over HTTP |
| [JGroups](http://camel.apache.org/jgroups.html) / camel-jgroups   |  | | --- | | jgroups:clusterName[?options] | | | The jgroups: component provides exchange of messages between Camel infrastructure and [JGroups](http://jgroups.org) clusters. |
| [JIRA](http://camel.apache.org/jira.html) / camel-jira   |  | | --- | | <jira://endpoint>[?options] | | | For interacting with JIRA |
| [JMS](http://camel.apache.org/jms.html) / camel-jms   |  | | --- | | jms:[queue:|topic:]destinationName[?options] | | | Working with JMS providers |
| [JMX](http://camel.apache.org/jmx.html) / camel-jmx   |  | | --- | | <jmx://platform>[?options] | | | For working with JMX notification listeners |
| [JPA](http://camel.apache.org/jpa.html) / camel-jpa   |  | | --- | | <jpa://entityName>[?options] | | | For using a database as a queue via the JPA specification for working with [OpenJPA](http://openjpa.apache.org/), [Hibernate](http://www.hibernate.org/) or TopLink |
| [JOLT](http://camel.apache.org/jolt.html) / camel-jolt   |  | | --- | | jolt:specName[?options] | | | The **jolt:** component allows you to process a JSON messages using an [JOLT](http://bazaarvoice.github.io/jolt/) specification. This can be ideal when doing JSON to JSON transformation. |
| [Jsch](http://camel.apache.org/jsch.html) / camel-jsch   |  | | --- | | <scp://hostName>[:port]/destination[?options] | | | Support for the scp protocol |
| [JT/400](http://camel.apache.org/jt400.html) / camel-jt400   |  | | --- | | [jt400://user:pwd@system/<path\_to\_dtaq](jt400://user:pwd@system/%3Cpath_to_dtaq)>[?options] | | | For integrating with data queues on an AS/400 (aka System i, IBM i, i5, ...) system |
| [Kafka](http://camel.apache.org/kafka.html) / camel-kafka   |  | | --- | | <kafka://server:port>[?options] | | | For producing to or consuming from [Apache Kafka](http://kafka.apache.org/) message brokers. |
| [Kestrel](http://camel.apache.org/kestrel.html) / camel-kestrel   |  | | --- | | [kestrel://](NULL)[addresslist/]queueName[?options] | | | For producing to or consuming from [Kestrel](https://github.com/robey/kestrel) queues |
| [Krati](http://camel.apache.org/krati.html) / camel-krati   |  | | --- | | [krati://](NULL)[path to datastore/][?options] | | | For producing to or consuming to [Krati](https://github.com/jingwei/krati) datastores |
| [K](http://camel.apache.org/krati.html)[ura](http://camel.apache.org/kura.html) / camel-kura | | For deploying Camel OSGi routes into the [Eclipse Kura](https://eclipse.org/kura/) M2M container. |
| [Language](http://camel.apache.org/language.html) / camel-core   |  | | --- | | <language://languageName>[:script][?options] | | | Executes [Languages](http://camel.apache.org/languages.html) scripts |
| [LDAP](http://camel.apache.org/ldap.html) / camel-ldap   |  | | --- | | ldap:host[:port][?options] | | | Performing searches on LDAP servers (<scope> must be one of object|onelevel|subtree) |
| [LinkedIn](http://camel.apache.org/linkedin.html) / camel-linkedin   |  | | --- | | <linkedin://endpoint-prefix/endpoint?>[options] | | | Component for retrieving LinkedIn user profiles, connections, companies, groups, posts, etc. using LinkedIn REST API. |
| [Log](http://camel.apache.org/log.html) / camel-core   |  | | --- | | log:loggingCategory[?options] | | | Uses Jakarta Commons Logging to log the message exchange to some underlying logging system like log4j |
| [Lucene](http://camel.apache.org/lucene.html) / camel-lucene   |  | | --- | | lucene:searcherName:<insert|query>[?options] | | | Uses Apache Lucene to perform Java-based indexing and full text based searches using advanced analysis/tokenization capabilities |
| [Metrics](http://camel.apache.org/metrics-component.html) / camel-metrics   |  | | --- | | metrics:[meter|counter|histogram|timer]:metricname[?options] | | | Uses [Metrics](http://metrics.codahale.com/)  to collect application statistics directly from Camel routes. |
| [MINA](http://camel.apache.org/mina.html) / camel-mina   |  | | --- | | mina:[tcp|udp|vm]:host[:port][?options] | | | Working with [Apache MINA 1.x](http://mina.apache.org/) |
| [MINA2](http://camel.apache.org/mina2.html) / camel-mina2   |  | | --- | | mina2:[tcp|udp|vm]:host[:port][?options] | | | Working with [Apache MINA 2.x](http://mina.apache.org/) |
| [Mock](http://camel.apache.org/mock.html) / camel-core   |  | | --- | | mock:name[?options] | | | For testing routes and mediation rules using mocks |
| [MongoDB](http://camel.apache.org/mongodb.html) / camel-mongodb   |  | | --- | | mongodb:connectionBean[?options] | | | Interacts with [MongoDB](http://www.mongodb.org/) databases and collections. Offers producer endpoints to perform CRUD-style operations and more against databases and collections, as well as consumer endpoints to listen on collections and dispatch objects to Camel routes |
| [MQTT](http://camel.apache.org/mqtt.html) / camel-mqtt   |  | | --- | | mqtt:name[?options] | | | Component for communicating with [MQTT](http://mqtt.org) M2M message brokers |
| [MSV](http://camel.apache.org/msv.html) / camel-msv   |  | | --- | | msv:someLocalOrRemoteResource[?options] | | | Validates the payload of a message using the [MSV Library](https://msv.java.net/) |
| [Mustache](http://camel.apache.org/mustache.html) / camel-mustache   |  | | --- | | mustache:templateName[?options] | | | Generates a response using a [Mustache](http://mustache.github.io/) template |
| [MVEL](http://camel.apache.org/mvel-component.html) / camel-mvel   |  | | --- | | mvel:templateName[?options] | | | Generates a response using an [MVEL](http://mvel.codehaus.org/) template |
| [MyBatis](http://camel.apache.org/mybatis.html) / camel-mybatis   |  | | --- | | <mybatis://statementName>[?options] | | | Performs a query, poll, insert, update or delete in a relational database using [MyBatis](http://mybatis.org/) |
| [Nagios](http://camel.apache.org/nagios.html) / camel-nagios   |  | | --- | | <nagios://hostName>[:port][?options] | | | Sending passive checks to [Nagios](http://www.nagios.org/) using [JSendNSCA](http://code.google.com/p/jsendnsca/) |
| [Netty](http://camel.apache.org/netty.html) / camel-netty   |  | | --- | | netty:<tcp|udp>//host[:port][?options] | | | Working with TCP and UDP protocols using Java NIO based capabilities offered by the [Netty](http://netty.io/) project |
| [Netty4](http://camel.apache.org/netty4.html) / camel-netty4   |  | | --- | | netty4:<tcp|udp>//host[:port][?options] | | | Working with TCP and UDP protocols using Java NIO based capabilities offered by the [Netty](http://netty.io/) project |
| [Netty HTTP](http://camel.apache.org/netty-http.html) / camel-netty-http   |  | | --- | | netty-http:http:[port]/context-path[?options] | | | Netty HTTP server and client using the [Netty](http://netty.io/) project |
| [Netty4 HTTP](http://camel.apache.org/netty4-http.html) / camel-netty4-http   |  | | --- | | netty4-http:http:[port]/context-path[?options] | | | Netty HTTP server and client using the [Netty](http://netty.io/) project 4.x |
| [Olingo2](http://camel.apache.org/olingo2.html) / camel-olingo2   |  | | --- | | olingo2:endpoint/resource-path[?options] | | | Communicates with [OData 2.0](http://www.odata.org/documentation/odata-version-2-0) services using [Apache Olingo](http://olingo.apache.org/) 2.0. |
| [O](http://camel.apache.org/optaplanner.html)[penshift](http://camel.apache.org/openshift.html) / camel-openshift   |  | | --- | | openshift:clientId[?options] | | | To manage your [Openshift](https://www.openshift.com/) applications. |
| [OptaPlanner](http://camel.apache.org/optaplanner.html) / camel-optaplanner   |  | | --- | | optaplanner:solverConfig[?options] | | | Solves the planning problem contained in a message with [OptaPlanner](http://www.optaplanner.org/). |
| [Paho](http://camel.apache.org/paho.html) / camel-paho   |  | | --- | | paho:topic[?options] | | | Paho component provides connector for the MQTT messaging protocol using the [Paho](https://eclipse.org/paho/) library. |
| [Pax-Logging](http://camel.apache.org/pax-logging.html) / camel-paxlogging   |  | | --- | | paxlogging:appender | | | Receiving Pax-Logging events in OSGi |
| [PDF](http://camel.apache.org/pdf.html) / camel-pdf   |  | | --- | | pdf:operation[?options] | | | Allows to work with Apache [PDFBox](https://pdfbox.apache.org/) PDF documents |
| [PGEvent](http://camel.apache.org/pgevent.html) / camel-pgevent   |  | | --- | | pgevent:dataSource[?options] | | | Allows for Producing/Consuming PostgreSQL events related to the LISTEN/NOTIFY commands added since PostgreSQL 8.3 |
| [POP3](http://camel.apache.org/mail.html) / camel-mail   |  | | --- | | [pop3s://](NULL)[username@]hostName port][?options] | | Receiving email using POP3 and JavaMail | |
| [POP3S](http://camel.apache.org/mail.html) / camel-mail   |  | | --- | | [pop3s://](NULL)[username@]hostName port][?options] | | ... | |
| [Printer](http://camel.apache.org/printer.html) / camel-printer   |  | | --- | | <lpr://host:port/path/to/printer>[?options] | | The printer component facilitates creation of printer endpoints to local, remote and wireless printers. The endpoints provide the ability to print camel directed payloads when utilized on camel routes. | |
| [Properties](http://camel.apache.org/properties.html) / camel-core   |  | | --- | | <properties://key>[?options] | | The properties component facilitates using property placeholders directly in endpoint uri definitions. | |
| [Quartz](http://camel.apache.org/quartz.html) / camel-quartz   |  | | --- | | <quartz://groupName/timerName>[?options] | | Provides a scheduled delivery of messages using the [Quartz 1.x scheduler](http://www.quartz-scheduler.org/) | |
| [Quartz2](http://camel.apache.org/quartz2.html) / camel-quartz2   |  | | --- | | <quartz2://groupName/timerName>[?options] | | Provides a scheduled delivery of messages using the [Quartz 2.x scheduler](http://www.quartz-scheduler.org/) | |
| [Quickfix](http://camel.apache.org/quickfix.html) / camel-quickfix   |  | | --- | | quickfix:configFile[?options] | | Implementation of the QuickFix for Java engine which allow to send/receive [FIX](http://www.fixprotocol.org) messages | |
| [RabbitMQ](http://camel.apache.org/rabbitmq.html) / camel-rabbitmq   |  | | --- | | <rabbitmq://hostname>[:port]/exchangeName[?options] | | Component for integrating with RabbitMQ | |
| [Ref](http://camel.apache.org/ref.html) / camel-core   |  | | --- | | ref:name | | Component for lookup of existing endpoints bound in the [Registry](http://camel.apache.org/registry.html). | |
| [Rest](http://camel.apache.org/rest.html) / camel-core   |  | | --- | | rest:verb:path[?options] | | Component for consuming Restful resources supporting the [Rest DSL](http://camel.apache.org/component-list.html) and plugins to other Camel rest components. | |
| [Restlet](http://camel.apache.org/restlet.html) / camel-restlet   |  | | --- | | restlet:restletUrl[?options] | | Component for consuming and producing Restful resources using [Restlet](http://www.restlet.org) | |
| [RMI](http://camel.apache.org/rmi.html) / camel-rmi   |  | | --- | | <rmi://hostName>[:port][?options] | | Working with RMI | |
| [RNC](http://camel.apache.org/jing.html) / camel-jing   |  | | --- | | rnc:/relativeOrAbsoluteUri[?options] | | Validates the payload of a message using [RelaxNG Compact Syntax](http://relaxng.org/compact-tutorial-20030326.html) | |
| [RNG](http://camel.apache.org/jing.html) / camel-jing   |  | | --- | | rng:/relativeOrAbsoluteUri[?options] | | Validates the payload of a message using [RelaxNG](http://relaxng.org/) | |
| [Routebox](http://camel.apache.org/routebox.html) / camel-routebox   |  | | --- | | routebox:routeBoxName[?options] | | Facilitates the creation of specialized endpoints that offer encapsulation and a strategy/map based indirection service to a collection of camel routes hosted in an automatically created or user injected camel context | |
| [RSS](http://camel.apache.org/rss.html) / camel-rss   |  | | --- | | rss:uri[?options] | | Working with [ROME](http://rometools.org/) for RSS integration, such as consuming an RSS feed. | |
| [Salesforce](http://camel.apache.org/salesforce.html) / camel-salesforce   |  | | --- | | salesforce:topic[?options] | | To integrate with Salesforce | |
| [SAP NetWeaver](http://camel.apache.org/sap-netweaver.html) / camel-sap-netweaver   |  | | --- | | sap-netweaver:hostName[:port][?options] | | To integrate with [SAP NetWeaver Gateway](http://scn.sap.com/docs/DOC-31221) | |
| [Scheduler](http://camel.apache.org/scheduler.html) / camel-core   |  | | --- | | <scheduler://name?>[options] | | Used to generate message exchanges when a scheduler fires. The scheduler has more functionality than the [timer](http://camel.apache.org/timer.html) component. | |
| [schematron](http://camel.apache.org/schematron.html) / camel-schematron   |  | | --- | | <schematron://path?>[options] | | Camel component of [Schematron](http://www.schematron.com/index.html) which supports to validate the XML instance documents. | |
| [SEDA](http://camel.apache.org/seda.html) / camel-core   |  | | --- | | seda:someName[?options] | | Asynchronous call to another endpoint in the same Camel Context | |
| [SERVLET](http://camel.apache.org/servlet.html) / camel-servlet   |  | | --- | | servlet:relativePath[?options] | | For exposing services over HTTP through the servlet which is deployed into the Web container. | |
| [SFTP](http://camel.apache.org/ftp2.html) / camel-ftp   |  | | --- | | [sftp://](NULL)[username@]hostName[:port]/directoryName[?options] | | Sending and receiving files over SFTP (FTP over SSH). | |
| [Sip](http://camel.apache.org/sip.html) / camel-sip   |  | | --- | | <sip://user@hostName>[:port][?options] | | Publish/Subscribe communication capability using the Telecom SIP protocol. [RFC3903 - Session Initiation Protocol (SIP) Extension for Event](http://www.ietf.org/rfc/rfc3903.txt) | |
| [SIPS](http://camel.apache.org/components.html) / camel-sip   |  | | --- | | <sips://user@hostName>[:port][?options] | | ... | |
| [SJMS](http://camel.apache.org/sjms.html)  / camel-sjms   |  | | --- | | sjms:[queue:|topic:]destinationName[?options] | | A ground up implementation of a JMS client | |
| [SJMS Batch](http://camel.apache.org/sjms-batch.html) / camel-sjms   |  | | --- | | sjms-batch:[queue:]destinationName[?options] | | A specialized JMS component for highly-performant transactional batch consumption from a queue. | |
| [Slack](http://camel.apache.org/slack.html) / camel-slack   |  | | --- | | slack:#channel[?options] | | The **slack** component allows you to connect to an instance of [Slack](http://www.slack.com/) and delivers a message contained in the message body via a pre established [Slack incoming webhook](https://api.slack.com/incoming-webhooks). | |
| [SMTP](http://camel.apache.org/mail.html) / camel-mail   |  | | --- | | [smtps://](NULL)[username@]hostName[:port][?options] | | Sending email using SMTP and JavaMail | |
| [SMTP](http://camel.apache.org/mail.html) / camel-mail   |  | | --- | | [smtps://](NULL)[username@]hostName[:port][?options] | | ... | |
| [SMPP](http://camel.apache.org/smpp.html) / camel-smpp   |  | | --- | | [smpp://](NULL)[username@]hostName[:port][?options] | | To send and receive SMS using Short Messaging Service Center using the [JSMPP library](http://code.google.com/p/jsmpp/) | |
| [SMPPS](http://camel.apache.org/components.html) / camel-smpp   |  | | --- | | [smpps://](NULL)[username@]hostName[:port][?options] | | ... | |
| [SNMP](http://camel.apache.org/snmp.html) / camel-snmp   |  | | --- | | <snmp://hostName>[:port][?options] | | Polling OID values and receiving traps using SNMP via [SNMP4J](http://snmp4j.com) library | |
| [Solr](http://camel.apache.org/solr.html) / camel-solr   |  | | --- | | <solr://hostName>[:port]/solr[?options] | | Uses the [Solrj](http://wiki.apache.org/solr/Solrj) client API to interface with an [Apache Lucene Solr](http://lucene.apache.org/solr/) server | |
| [Spark-rest](http://camel.apache.org/spark-rest.html) / camel-spark-rest   |  | | --- | | spark-<rest://verb:path?>[options] | | For easily defining REST services endpoints using [Spark REST Java](http://sparkjava.com/) library. | |
| [Splunk](http://camel.apache.org/splunk.html) / camel-splunk   |  | | --- | | [splunk://](NULL)[endpoint]?[options] | | For working with [Splunk](http://docs.splunk.com/Documentation/Splunk) | |
| [SpringBatch](http://camel.apache.org/springbatch.html) / camel-spring-batch   |  | | --- | | spring-<batch://jobName>[?options] | | To bridge Camel and [Spring Batch](http://www.springsource.org/spring-batch) | |
| [SpringIntegration](http://camel.apache.org/springintegration.html) / camel-spring-integration   |  | | --- | | spring-integration:defaultChannelName[?options] | | The bridge component of Camel and [Spring Integration](http://www.springframework.org/spring-integration) | |
| [Spring LDAP](http://camel.apache.org/spring-ldap.html) / camel-spring-ldap   |  | | --- | | spring-ldap:springLdapTemplateBean[?options] | | Camel wrapper for [Spring LDAP](http://www.springsource.org/ldap) | |
| [Spring Redis](http://camel.apache.org/spring-redis.html) / camel-spring-redis   |  | | --- | | spring-<redis://hostName:port>[?options] | | Component for consuming and producing from Redis key-value store [Redis](http://redis.io) | |
| [Spring Web Services](http://camel.apache.org/spring-web-services.html) / camel-spring-ws   |  | | --- | | spring-ws:[mapping-type:]address[?options] | | Client-side support for accessing web services, and server-side support for creating your own contract-first web services using [Spring Web Services](http://static.springsource.org/spring-ws/sites/1.5/) | |
| [SQL](http://camel.apache.org/sql-component.html) / camel-sql   |  | | --- | | sql:select \* from table where id=#[?options] | | Performing SQL queries using JDBC | |
| [SSH](http://camel.apache.org/ssh.html) component / camel-ssh   |  | | --- | | ssh:[username[:password]@]hostName[:port][?options] | | For sending commands to a SSH server | |
| [StAX](http://camel.apache.org/stax.html) / camel-stax   |  | | --- | | stax:(contentHandlerClassName|#myHandler) | | Process messages through a SAX [ContentHandler](http://download.oracle.com/javase/6/docs/api/org/xml/sax/ContentHandler.html). | |
| [Stream](http://camel.apache.org/stream.html) / camel-stream   |  | | --- | | stream:[in|out|err|file|header|url][?options] | | Read or write to an input/output/error/file stream rather like unix pipes | |
| [Stomp](http://camel.apache.org/stomp.html) / camel-stomp   |  | | --- | | stomp:queue:destinationName[?options] | | For communicating with [Stomp](http://stomp.github.io/) compliant message brokers, like [Apache ActiveMQ](http://activemq.apache.org) or [ActiveMQ Apollo](http://activemq.apache.org/apollo/) | |
| [StringTemplate](http://camel.apache.org/stringtemplate.html) / camel-stringtemplate   |  | | --- | | string-template:templateName[?options] | | Generates a response using a [String Template](http://www.stringtemplate.org/) | |
| [Stub](http://camel.apache.org/stub.html) / camel-core   |  | | --- | | stub:someOtherCamelUri[?options] | | Allows you to [stub out some physical middleware endpoint](http://camel.apache.org/stub.html) for easier testing or debugging | |
| [Test](http://camel.apache.org/test.html) / camel-spring   |  | | --- | | test:expectedMessagesEndpointUri[?options] | | Creates a [Mock](http://camel.apache.org/mock.html) endpoint which expects to receive all the message bodies that could be polled from the given underlying endpoint | |
| [Timer](http://camel.apache.org/timer.html) / camel-core   |  | | --- | | timer:timerName[?options] | | Used to generate message exchanges when a timer fires You can only consume events from this endpoint. | |
| [Twitter](http://camel.apache.org/twitter.html) / camel-twitter   |  | | --- | | <twitter://endpoint>[?options] | | A twitter endpoint | |
| [Undertow](http://camel.apache.org/undertow.html) / camel-undertow   |  | | --- | | <undertow://host:port/context-path>[?options] | | HTTP server and client using the light-weight [Undertow](http://undertow.io/index.html) server. | |
| [Validation](http://camel.apache.org/validation.html) / camel-core (camel-spring for Camel 2.8 or older)   |  | | --- | | validation:someLocalOrRemoteResource[?options] | | Validates the payload of a message using [XML Schema](http://www.w3.org/XML/Schema) and JAXP Validation | |
| [Velocity](http://camel.apache.org/velocity.html) / camel-velocity   |  | | --- | | velocity:templateName[?options] | | Generates a response using an [Apache Velocity](http://velocity.apache.org/) template | |
| [Vertx](http://camel.apache.org/vertx.html) / camel-vertx   |  | | --- | | vertx:eventBusName | | Working with the [vertx](http://vertx.io/) event bus | |
| [VM](http://camel.apache.org/vm.html) / camel-core   |  | | --- | | vm:queueName[?options] | | Asynchronous call to another endpoint in the same JVM | |
| [Weather](http://camel.apache.org/weather.html) / camel-weather   |  | | --- | | <wweather://name>[?options] | | Polls the weather information from [Open Weather Map](http://openweathermap.org) | |
| [Websocket](http://camel.apache.org/websocket.html) / camel-websocket   |  | | --- | | <websocket://hostname>[:port][/resourceUri][?options] | | Communicating with [Websocket](http://wiki.eclipse.org/Jetty/Feature/WebSockets) clients | |
| [XML Security](http://camel.apache.org/xml-security-component.html) / camel-xmlsecurity   |  | | --- | | xmlsecurity:<sign|verify>:name[?options] | | Used to sign and verify exchanges using the XML signature specification. | |
| [XMPP](http://camel.apache.org/xmpp.html) / camel-xmpp   |  | | --- | | [xmpp://](NULL)[login@]hostname[:port][/participant][?options] | | Working with XMPP and Jabber | |
| [XQuery](http://camel.apache.org/xquery-endpoint.html) / camel-saxon   |  | | --- | | xquery:someXQueryResource | | Generates a response using an [XQuery](http://camel.apache.org/xquery.html) template | |
| [XSLT](http://camel.apache.org/xslt.html) / camel-core (camel-spring for Camel 2.8 or older)   |  | | --- | | xslt:templateName[?options] | | Generates a response using an [XSLT](http://www.w3.org/TR/xslt) template | |
| [Yammer](http://camel.apache.org/yammer.html) / camel-yammer   |  | | --- | | <yammer://function>[?options] | | Allows you to interact with the [Yammer](http://yammer.com) enterprise social network | |
| [Zookeeper](http://camel.apache.org/zookeeper.html) / camel-zookeeper   |  | | --- | | <zookeeper://zookeeperServer>[:port][/path][?options] | | Working with [ZooKeeper](http://hadoop.apache.org/zookeeper/) cluster(s) | |

**External Components**

The following components are not part of the standard Apache Camel distribution and are available under a variety of licenses but can be used to extend Camel's functionality.

|  |  |  |
| --- | --- | --- |
| **Component / ArtifactId / URI** | **License** | **Description** |
| [ActiveMQ](http://camel.apache.org/activemq.html) / activemq-camel   |  | | --- | | activemq:[queue|topic:]destinationName | | Apache | For JMS Messaging with [Apache ActiveMQ](http://activemq.apache.org/) |
| [ActiveMQ Broker](http://activemq.apache.org/broker-camel-component.html) / activemq-camel   |  | | --- | | broker:[queue|topic:]destinationName | | Apache | For internal message routing in the [ActiveMQ](http://camel.apache.org/activemq.html) broker using Camel. |
| [Activiti](http://activiti.org/userguide/index.html#bpmnCamelTask) / activiti-camel   |  | | --- | | activiti:camelProcess:serviceTask | | Apache | For working with [Activiti](http://www.activiti.org/), a light-weight workflow and Business Process Management (BPM) platform which supports BPMN 2 |
| [Couchbase](http://camel.apache.org/couchbase.html) / camel-couchbase in [camel-extra](https://code.google.com/a/apache-extras.org/p/camel-extra/)   |  | | --- | | couchbase:<protocol://host>[:port]/bucket | | Couchbase | Working with Couchbase NoSQL document database |
| [Db4o](http://camel.apache.org/db4o.html) / camel-db4o in [camel-extra](http://code.google.com/p/camel-extra/)   |  | | --- | | <db4o://className> | | GPL | For using a db4o datastore as a queue via the [db4o](http://www.db4o.com/) library |
| [Esper](http://camel.apache.org/esper.html) / camel-esper in [camel-extra](http://code.google.com/p/camel-extra/)   |  | | --- | | esper:name | | GPL | Working with the [Esper Library](http://esper.codehaus.org) for Event Stream Processing |
| Fabric [AMQ](http://fabric8.io/gitbook/camelEndpointAmq.html) / mq-fabric-camel in [f](http://code.google.com/p/camel-extra/)[abric8](http://fabric8.io/)   |  | | --- | | amq:[queue|topic:]destinationName | | Apache | The [**amq:**](http://fabric8.io/gitbook/camelEndpointAmq.html) endpoint works exactly like the **activemq:** endpoint in Apache Camel; only it uses the [fabric](http://fabric8.io/) to automatically discover the broker. So there is no configuration required; it'll just work out of the box and automatically discover whatever ActiveMQ message brokers are available; with failover and load balancing. |
| Fabric [Fabric](http://fabric8.io/gitbook/camelEndpointFabric.html) / fabric-camel in [f](http://code.google.com/p/camel-extra/)[abric8](http://fabric8.io/)   |  | | --- | | fabric:logicalName:camelEndpointUri | | Apache | The [**fabric:**](http://fabric8.io/gitbook/camelEndpointFabric.html) endpoint uses Fabric's discovery mechanism to expose physical sockets, HTTP endpoints, etc. into the [runtime registry](http://fabric8.io/gitbook/registry.html) using a logical name so that clients can use the existing Camel [Load Balancer](http://camel.apache.org/load-balancer.html). |
| Fabric [Master](http://fabric8.io/gitbook/camelEndpointMaster.html) / fabric-camel in [f](http://code.google.com/p/camel-extra/)[abric8](http://fabric8.io/)   |  | | --- | | master:clusterName:camelEndpointUri | | Apache | The [**master:**](http://fabric8.io/gitbook/camelEndpointMaster.html) endpoint provides a way to ensure only a single consumer in a cluster consumes from a given endpoint; with automatic failover if that JVM dies. |
| [Hibernate](http://camel.apache.org/hibernate.html) / camel-hibernate in [camel-extra](http://code.google.com/p/camel-extra/)   |  | | --- | | <hibernate://entityName> | | GPL | For using a database as a queue via the [Hibernate](http://www.hibernate.org/) library |
| [JBI](http://camel.apache.org/jbi.html) / servicemix-camel   |  | | --- | | jbi:serviceName | | Apache | For JBI integration such as working with [Apache ServiceMix](http://servicemix.apache.org) |
| [JCIFS](http://camel.apache.org/jcifs.html) / camel-jcifs in [camel-extra](http://code.google.com/p/camel-extra/)   |  | | --- | | <smb://user@server.example.com/sharename?password=secret&localWorkDirectory=/tmp> | | LGPL | This component provides access to remote file systems over the CIFS/SMB networking protocol by using the [JCIFS](http://jcifs.samba.org/) library. |
| [NMR](http://camel.apache.org/nmr.html) / servicemix-nmr   |  | | --- | | <nmr://serviceName> | | Apache | Integration with the Normalized Message Router BUS in [ServiceMix 4.x](http://servicemix.apache.org/SMX4NMR/index.html) |
| [pi4j-gpio](http://camel.apache.org/pi4j-gpio.html) / camel-pi4j in [Camel IoT Labs](https://github.com/camel-labs/camel-labs/tree/master/iot)   |  | | --- | | pi4j-<gpio://gpioId>[?options] | | Apache | GPIO Component for RaspberryPi |
| [pi4j-i2c](https://github.com/camel-labs/camel-labs/tree/master/iot) / camel-pi4j in [Camel IoT Labs](https://github.com/camel-labs/camel-labs/tree/master/iot)   |  | | --- | | pi4j-<i2c://busId/deviceId>[?options] | | Apache | i2c Component for RaspberryPi |
| [PubNub](https://github.com/camel-labs/camel-labs/tree/master/iot#camel-pubnub-component) / camel-pubnub in [Camel IoT Labs](https://github.com/camel-labs/camel-labs/tree/master/iot)   |  | | --- | | <pubnub://pubnubEndpointType:channel>[?options] | | Apache | Camel [PubNub](https://www.pubnub.com/) component. |
| [RCode](http://camel.apache.org/rcode.html) / camel-rcode in [camel-extra](http://code.google.com/p/camel-extra/)   |  | | --- | | <rcode://host>[:port]/operation[?options] | | LGPL | Uses [Rserve](http://www.rforge.net/Rserve/) to integrate Camel with the statistics environment [R](http://www.r-project.org/) |
| [Scalate](http://scalate.fusesource.org/camel.html) / scalate-camel   |  | | --- | | scalate:templateName | | Apache | Uses the given [Scalate](http://scalate.fusesource.org/) template to transform the message |
| [Smooks](http://camel.apache.org/smooks.html) / camel-smooks in [camel-extra](http://code.google.com/p/camel-extra/).   |  | | --- | | unmarshal(edi) | | GPL | For working with EDI parsing using the [Smooks library](http://milyn.codehaus.org/Smooks). This component is **deprecated** as Smooks now provides [Camel integration out of the box](http://www.smooks.org/mediawiki/index.php?title=V1.5:Smooks_v1.5_User_Guide#Apache_Camel_Integration) |
| [Spring Neo4j](http://camel.apache.org/spring-neo4j.html) / camel-spring-neo4j in [camel-extra](http://code.google.com/p/camel-extra/)   |  | | --- | | spring-neo4j:<http://hostname>[:port]/database[?options] | | to bee clarified | Component for producing to Neo4j datastore using the [Spring Data Neo4j](http://www.springsource.org/spring-data/neo4j) library |
| [Tinkerforge](http://www.tinkerforge.com) / camel-tinkerforge in [Camel IoT Labs](https://github.com/camel-labs/camel-labs/tree/master/iot)   |  | | --- | | tinkerforge:[//hostname[:port]]/devicetype/uid/[?options] | | Apache | The tinkerforge component allows interaction with Tinkerforge [bricklets](http://www.tinkerforge.com/en/doc/Primer.html#primer-bricklets). It uses the standard [Java bindings](http://www.tinkerforge.com/en/doc/Software/API_Bindings_Java.html#api-bindings-java) to connects to brickd. |
| [VirtualBox](http://camel.apache.org/virtualbox.html) / camel-virtualbox in [camel-extra](http://code.google.com/p/camel-extra/).   |  | | --- | | virtualbox:machine[?options] | | GPL V2 | The VitualBox component uses the webservice API that exposes [VirtualBox](https://www.virtualbox.org/wiki/VirtualBox) functionality and consumes events generated by virtual machines. |
| [ZeroMQ](http://camel.apache.org/zeromq.html) / camel-zeromq in [camel-extra](http://code.google.com/p/camel-extra/).   |  | | --- | | zeromq:(tcp|ipc)://hostname:port | | LGPL | The ZeroMQ component allows you to consumer or produce messages using [ZeroMQ](http://zeromq.org). |

## Using PropertyPlaceholder

**Available as of Camel 2.3**

Camel now provides a new PropertiesComponent in **camel-core** which allows you to use property placeholders when defining Camel [Endpoint](http://camel.apache.org/endpoint.html) URIs.   
This works much like you would do if using Spring's <property-placeholder> tag. However Spring have a limitation which prevents 3rd party frameworks to leverage Spring property placeholders to the fullest. See more at [How do I use Spring Property Placeholder with Camel XML](http://camel.apache.org/how-do-i-use-spring-property-placeholder-with-camel-xml.html).

Bridging Spring and Camel property placeholders

From Camel 2.10 onwards, you can bridge the Spring property placeholder with Camel, see further below for more details.

The property placeholder is generally in use when doing:

* lookup or creating endpoints
* lookup of beans in the [Registry](http://camel.apache.org/registry.html)
* additional supported in Spring XML (see below in examples)
* using Blueprint PropertyPlaceholder with Camel [Properties](http://camel.apache.org/properties.html) component
* using @PropertyInject to inject a property in a POJO
* **Camel 2.14.1** Using default value if a property does not exists
* **Camel 2.14.1** Include out of the box functions, to lookup property values from OS environment variables, JVM system properties, or the service idiom.
* **Camel 2.14.1** Using custom functions, which can be plugged into the property component.

### Syntax

The syntax to use Camel's property placeholder is to use {{key}} for example {{file.uri}} where file.uri is the property key.  
You can use property placeholders in parts of the endpoint URI's which for example you can use placeholders for parameters in the URIs.

From **Camel 2.14.1** onwards you can specify a default value to use if a property with the key does not exists, eg file.url:/some/path where the default value is the text after the colon (eg /some/path).

Do not use colon in the property key. The colon is used as a separator token when you are providing a default value, which is supported from **Camel 2.14.1** onwards.

### PropertyResolver

Camel provides a pluggable mechanism which allows 3rd part to provide their own resolver to lookup properties. Camel provides a default implementation org.apache.camel.component.properties.DefaultPropertiesResolver which is capable of loading properties from the file system, classpath or [Registry](http://camel.apache.org/registry.html). You can prefix the locations with either:

* ref: **Camel 2.4:** to lookup in the [Registry](http://camel.apache.org/registry.html)
* file: to load the from file system
* classpath: to load from classpath (this is also the default if no prefix is provided)
* blueprint: **Camel 2.7:** to use a specific OSGi blueprint placeholder service

### Defining location

The PropertiesResolver need to know a location(s) where to resolve the properties. You can define 1 to many locations. If you define the location in a single String property you can separate multiple locations with comma such as:

|  |
| --- |
| pc.setLocation("com/mycompany/myprop.properties,com/mycompany/other.properties"); |

#### Using system and environment variables in locations

**Available as of Camel 2.7**

The location now supports using placeholders for JVM system properties and OS environments variables.

For example:

|  |
| --- |
| location=file:${karaf.home}/etc/foo.properties |

In the location above we defined a location using the file scheme using the JVM system property with key karaf.home.

To use an OS environment variable instead you would have to prefix with env:

|  |
| --- |
| location=file:${env:APP\_HOME}/etc/foo.properties |

Where APP\_HOME is an OS environment.

You can have multiple placeholders in the same location, such as:

|  |
| --- |
| location=file:${env:APP\_HOME}/etc/${prop.name}.properties |

#### Using system and environment variables to configure property prefixes and suffixes

**Available as of Camel 2.12.5, 2.13.3, 2.14.0**

propertyPrefix, propertySuffix configuration properties support using placeholders for JVM system properties and OS environments variables.

For example. if PropertiesComponent is configured with the following properties file:

dev.endpoint = result1

test.endpoint = result2

Then with the following route definition:

|  |
| --- |
| PropertiesComponent pc = context.getComponent("properties", PropertiesComponent.class);  pc.setPropertyPrefix("${stage}.");  // ...  context.addRoutes(new RouteBuilder() {      @Override      public void configure() throws Exception {          from("direct:start").to("properties:mock:{{endpoint}}");      }  }); |

it is possible to change the target endpoint by changing system property stage either to dev (the message will be routed to mock:result1) or test (the message will be routed to mock:result2).

### Configuring in Java DSL

You have to create and register the PropertiesComponent under the name properties such as:

|  |
| --- |
| PropertiesComponent pc = new PropertiesComponent();  pc.setLocation("classpath:com/mycompany/myprop.properties");  context.addComponent("properties", pc); |

### Configuring in Spring XML

Spring XML offers two variations to configure. You can define a spring bean as a PropertiesComponent which resembles the way done in Java DSL. Or you can use the <propertyPlaceholder> tag.

|  |
| --- |
| <bean id="properties" class="org.apache.camel.component.properties.PropertiesComponent">      <property name="location" value="classpath:com/mycompany/myprop.properties"/>  </bean> |

Using the <propertyPlaceholder> tag makes the configuration a bit more fresh such as:

|  |
| --- |
| <camelContext ...>     <propertyPlaceholder id="properties" location="com/mycompany/myprop.properties"/>  </camelContext> |

Specifying the cache option inside XML

Camel 2.10 onwards supports specifying a value for the cache option both inside the Spring as well as the Blueprint XML.

### Using a Properties from the [Registry](http://camel.apache.org/registry.html)

**Available as of Camel 2.4**  
For example in OSGi you may want to expose a service which returns the properties as a java.util.Properties object.

Then you could setup the [Properties](http://camel.apache.org/properties.html) component as follows:

|  |
| --- |
| <propertyPlaceholder id="properties" location="ref:myProperties"/> |

Where myProperties is the id to use for lookup in the OSGi registry. Notice we use the ref: prefix to tell Camel that it should lookup the properties for the [Registry](http://camel.apache.org/registry.html).

### Examples using properties component

When using property placeholders in the endpoint URIs you can either use the properties: component or define the placeholders directly in the URI. We will show example of both cases, starting with the former.

|  |
| --- |
| // properties  cool.end=mock:result    // route  from("direct:start").to("properties:{{cool.end}}"); |

You can also use placeholders as a part of the endpoint uri:

|  |
| --- |
| // properties  cool.foo=result    // route  from("direct:start").to("properties:mock:{{cool.foo}}"); |

In the example above the to endpoint will be resolved to mock:result.

You can also have properties with refer to each other such as:

|  |
| --- |
| // properties  cool.foo=result  cool.concat=mock:{{cool.foo}}    // route  from("direct:start").to("properties:mock:{{cool.concat}}"); |

Notice how cool.concat refer to another property.

The properties: component also offers you to override and provide a location in the given uri using the locations option:

|  |
| --- |
| from("direct:start").to("properties:bar.end?locations=com/mycompany/bar.properties"); |

### Examples

You can also use property placeholders directly in the endpoint uris without having to use properties:.

|  |
| --- |
| // properties  cool.foo=result    // route  from("direct:start").to("mock:{{cool.foo}}"); |

And you can use them in multiple wherever you want them:

|  |
| --- |
| // properties  cool.start=direct:start  cool.showid=true  cool.result=result    // route  from("{{cool.start}}")      .to("log:{{cool.start}}?showBodyType=false&showExchangeId={{cool.showid}}")      .to("mock:{{cool.result}}"); |

You can also your property placeholders when using [ProducerTemplate](http://camel.apache.org/producertemplate.html) for example:

|  |
| --- |
| template.sendBody("{{cool.start}}", "Hello World"); |

### Example with [Simple](http://camel.apache.org/simple.html) language

The [Simple](http://camel.apache.org/simple.html) language now also support using property placeholders, for example in the route below:

|  |
| --- |
| // properties  cheese.quote=Camel rocks    // route  from("direct:start")      .transform().simple("Hi ${body} do you think ${properties:cheese.quote}?"); |

You can also specify the location in the [Simple](http://camel.apache.org/simple.html) language for example:

|  |
| --- |
| // bar.properties  bar.quote=Beer tastes good    // route  from("direct:start")      .transform().simple("Hi ${body}. ${properties:com/mycompany/bar.properties:bar.quote}."); |

### Additional property placeholder supported in Spring XML

The property placeholders is also supported in many of the Camel Spring XML tags such as <package>, <packageScan>, <contextScan>, <jmxAgent>, <endpoint>, <routeBuilder>, <proxy> and the others.

The example below has property placeholder in the <jmxAgent> tag:

|  |
| --- |
| <camelContext xmlns="[http://camel.apache.org/schema/spring"](http://camel.apache.org/schema/spring%22)>      <propertyPlaceholder id="properties" location="org/apache/camel/spring/jmx.properties"/>        <!-- we can use propery placeholders when we define the JMX agent -->      <jmxAgent id="agent" registryPort="{{myjmx.port}}" disabled="{{myjmx.disabled}}"                usePlatformMBeanServer="{{myjmx.usePlatform}}"                createConnector="true"                statisticsLevel="RoutesOnly"                useHostIPAddress="true"/>        <route id="foo" autoStartup="false">          <from uri="seda:start"/>          <to uri="mock:result"/>      </route>    </camelContext> |

You can also define property placeholders in the various attributes on the <camelContext> tag such as trace as shown here:

|  |
| --- |
| <camelContext trace="{{foo.trace}}" xmlns="[http://camel.apache.org/schema/spring"](http://camel.apache.org/schema/spring%22)>      <propertyPlaceholder id="properties" location="org/apache/camel/spring/processor/myprop.properties"/>        <template id="camelTemplate" defaultEndpoint="{{foo.cool}}"/>        <route>          <from uri="direct:start"/>          <setHeader headerName="{{foo.header}}">              <simple>${in.body} World!</simple>          </setHeader>          <to uri="mock:result"/>      </route>  </camelContext> |

### Overriding a property setting using a JVM System Property

**Available as of Camel 2.5**  
It is possible to override a property value at runtime using a JVM System property without the need to restart the application to pick up the change. This may also be accomplished from the command line by creating a JVM System property of the same name as the property it replaces with a new value. An example of this is given below

|  |
| --- |
| PropertiesComponent pc = context.getComponent("properties", PropertiesComponent.class);  pc.setCache(false);    System.setProperty("cool.end", "mock:override");  System.setProperty("cool.result", "override");    context.addRoutes(new RouteBuilder() {      @Override      public void configure() throws Exception {          from("direct:start").to("properties:cool.end");          from("direct:foo").to("properties:mock:{{cool.result}}");      }  });  context.start();    getMockEndpoint("mock:override").expectedMessageCount(2);    template.sendBody("direct:start", "Hello World");  template.sendBody("direct:foo", "Hello Foo");    System.clearProperty("cool.end");  System.clearProperty("cool.result");    assertMockEndpointsSatisfied(); |

### Using property placeholders for any kind of attribute in the XML DSL

**Available as of Camel 2.7**

If you use OSGi Blueprint then this only works from **2.11.1** or **2.10.5** onwards.

Previously it was only the xs:string type attributes in the XML DSL that support placeholders. For example often a timeout attribute would be a xs:int type and thus you cannot set a string value as the placeholder key. This is now possible from Camel 2.7 onwards using a special placeholder namespace.

In the example below we use the prop prefix for the namespace <http://camel.apache.org/schema/placeholder> by which we can use the prop prefix in the attributes in the XML DSLs. Notice how we use that in the [Multicast](http://camel.apache.org/multicast.html) to indicate that the option stopOnException should be the value of the placeholder with the key "stop".

|  |
| --- |
| <beans xmlns="[http://www.springframework.org/schema/beans"](http://www.springframework.org/schema/beans%22)         xmlns:xsi="[http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance%22)         xmlns:prop="[http://camel.apache.org/schema/placeholder"](http://camel.apache.org/schema/placeholder%22)         xsi:schemaLocation="  <http://www.springframework.org/schema/beans> <http://www.springframework.org/schema/beans/spring-beans.xsd>  <http://camel.apache.org/schema/spring> <http://camel.apache.org/schema/spring/camel-spring.xsd>      ">        <!-- Notice in the declaration above, we have defined the prop prefix as the Camel placeholder namespace -->        <bean id="damn" class="java.lang.IllegalArgumentException">          <constructor-arg index="0" value="Damn"/>      </bean>        <camelContext xmlns="[http://camel.apache.org/schema/spring"](http://camel.apache.org/schema/spring%22)>            <propertyPlaceholder id="properties"                               location="classpath:org/apache/camel/component/properties/myprop.properties"                               xmlns="[http://camel.apache.org/schema/spring"/](http://camel.apache.org/schema/spring%22/)>            <route>              <from uri="direct:start"/>              <!-- use prop namespace, to define a property placeholder, which maps to                   option stopOnException={{stop}} -->              <multicast prop:stopOnException="stop">                  <to uri="mock:a"/>                  <throwException ref="damn"/>                  <to uri="mock:b"/>              </multicast>          </route>        </camelContext>    </beans> |

In our properties file we have the value defined as

|  |
| --- |
| stop=true |

### Using property placeholder in the Java DSL

**Available as of Camel 2.7**

Likewise we have added support for defining placeholders in the Java DSL using the new placeholder DSL as shown in the following equivalent example:

|  |
| --- |
| from("direct:start")      // use a property placeholder for the option stopOnException on the Multicast EIP      // which should have the value of {{stop}} key being looked up in the properties file      .multicast().placeholder("stopOnException", "stop")          .to("mock:a").throwException(new IllegalAccessException("Damn")).to("mock:b"); |

### Using Blueprint property placeholder with Camel routes

**Available as of Camel 2.7**

Camel supports [Blueprint](http://camel.apache.org/using-osgi-blueprint-with-camel.html) which also offers a property placeholder service. Camel supports convention over configuration, so all you have to do is to define the OSGi Blueprint property placeholder in the XML file as shown below:

**Using OSGi blueprint property placeholders in Camel routes**

|  |
| --- |
| <blueprint xmlns="[http://www.osgi.org/xmlns/blueprint/v1.0.0"](http://www.osgi.org/xmlns/blueprint/v1.0.0%22)             xmlns:xsi="[http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance%22)             xmlns:cm="[http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"](http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0%22)             xsi:schemaLocation="  <http://www.osgi.org/xmlns/blueprint/v1.0.0> [http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd"](http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd%22)>        <!-- OSGI blueprint property placeholder -->      <cm:property-placeholder id="myblueprint.placeholder" persistent-id="camel.blueprint">          <!-- list some properties for this test -->          <cm:default-properties>              <cm:property name="result" value="mock:result"/>          </cm:default-properties>      </cm:property-placeholder>        <camelContext xmlns="[http://camel.apache.org/schema/blueprint"](http://camel.apache.org/schema/blueprint%22)>            <!-- in the route we can use {{ }} placeholders which will lookup in blueprint               as Camel will auto detect the OSGi blueprint property placeholder and use it -->          <route>              <from uri="direct:start"/>              <to uri="mock:foo"/>              <to uri="{{result}}"/>          </route>        </camelContext>    </blueprint> |

By default Camel detects and uses OSGi blueprint property placeholder service. You can disable this by setting the attribute useBlueprintPropertyResolver to false on the <camelContext> definition.

About placeholder syntaxes

Notice how we can use the Camel syntax for placeholders {{ }} in the Camel route, which will lookup the value from OSGi blueprint.  
The blueprint syntax for placeholders is ${ }. So outside the <camelContext> you must use the ${ } syntax. Where as inside <camelContext> you must use {{ }} syntax.  
OSGi blueprint allows you to configure the syntax, so you can actually align those if you want.

You can also explicit refer to a specific OSGi blueprint property placeholder by its id. For that you need to use the Camel's <propertyPlaceholder> as shown in the example below:

**Explicit referring to a OSGi blueprint placeholder in Camel**

|  |
| --- |
| <blueprint xmlns="[http://www.osgi.org/xmlns/blueprint/v1.0.0"](http://www.osgi.org/xmlns/blueprint/v1.0.0%22)             xmlns:xsi="[http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance%22)             xmlns:cm="[http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0"](http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.0.0%22)             xsi:schemaLocation="  <http://www.osgi.org/xmlns/blueprint/v1.0.0> [http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd"](http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd%22)>        <!-- OSGI blueprint property placeholder -->      <cm:property-placeholder id="myblueprint.placeholder" persistent-id="camel.blueprint">          <!-- list some properties for this test -->          <cm:default-properties>              <cm:property name="prefix.result" value="mock:result"/>          </cm:default-properties>      </cm:property-placeholder>        <camelContext xmlns="[http://camel.apache.org/schema/blueprint"](http://camel.apache.org/schema/blueprint%22)>            <!-- using Camel properties component and refer to the blueprint property placeholder by its id -->          <propertyPlaceholder id="properties" location="blueprint:myblueprint.placeholder"                               prefixToken="[[" suffixToken="]]"                               propertyPrefix="prefix."/>            <!-- in the route we can use {{ }} placeholders which will lookup in blueprint -->          <route>              <from uri="direct:start"/>              <to uri="mock:foo"/>              <to uri="[[result]]"/>          </route>        </camelContext>    </blueprint> |

Notice how we use the blueprint scheme to refer to the OSGi blueprint placeholder by its id. This allows you to mix and match, for example you can also have additional schemes in the location. For example to load a file from the classpath you can do:

|  |
| --- |
| location="blueprint:myblueprint.placeholder,classpath:myproperties.properties" |

Each location is separated by comma.

#### Overriding Blueprint property placeholders outside CamelContext

**Available as of Camel 2.10.4**

When using Blueprint property placeholder in the Blueprint XML file, you can declare the properties directly in the XML file as shown below:

|  |
| --- |
| <!-- blueprint property placeholders -->  <cm:property-placeholder persistent-id="my-placeholders" update-strategy="reload">    <cm:default-properties>      <cm:property name="greeting" value="Hello"/>      <cm:property name="destination" value="mock:result"/>    </cm:default-properties>  </cm:property-placeholder>    <!-- a bean that uses a blueprint property placeholder -->  <bean id="myCoolBean" class="org.apache.camel.test.blueprint.MyCoolBean">    <property name="say" value="${greeting}"/>  </bean>    <camelContext xmlns="[http://camel.apache.org/schema/blueprint"](http://camel.apache.org/schema/blueprint%22)>      <route>      <from uri="direct:start"/>      <bean ref="myCoolBean" method="saySomething"/>      <to uri="{{destination}}"/>    </route>    </camelContext> |

Notice that we have a <bean> which refers to one of the properties. And in the Camel route we refer to the other using the {{ }} notation.

Now if you want to override these Blueprint properties from an unit test, you can do this as shown below:

|  |
| --- |
| @Override  protected String useOverridePropertiesWithConfigAdmin(Dictionary props) {      // add the properties we want to override      props.put("greeting", "Bye");        // return the PID of the config-admin we are using in the blueprint xml file      return "my-placeholders";  } |

To do this we override and implement the useOverridePropertiesWithConfigAdmin method. We can then put the properties we want to override on the given props parameter. And the return value **must** be the persistence-id of the <cm:property-placeholder> tag, which you define in the blueprint XML file.

#### Using .cfg or .properties file for Blueprint property placeholders

**Available as of Camel 2.10.4**

When using Blueprint property placeholder in the Blueprint XML file, you can declare the properties in a .properties or .cfg file. If you use Apache ServieMix / Karaf then this container has a convention that it loads the properties from a file in the etc directory with the naming etc/pid.cfg, where pid is the persistence-id.

For example in the blueprint XML file we have the persistence-id="stuff", which mean it will load the configuration file as etc/stuff.cfg.

|  |
| --- |
| <!-- blueprint property placeholders, that will use etc/stuff.cfg as the properties file -->  <cm:property-placeholder persistent-id="stuff" update-strategy="reload"/>    <!-- a bean that uses a blueprint property placeholder -->  <bean id="myCoolBean" class="org.apache.camel.test.blueprint.MyCoolBean">    <property name="say" value="${greeting}"/>  </bean>    <camelContext xmlns="[http://camel.apache.org/schema/blueprint"](http://camel.apache.org/schema/blueprint%22)>      <route>      <from uri="direct:start"/>      <bean ref="myCoolBean" method="saySomething"/>      <to uri="mock:result"/>    </route>    </camelContext> |

Now if you want to unit test this blueprint XML file, then you can override the loadConfigAdminConfigurationFile and tell Camel which file to load as shown below:

|  |
| --- |
| @Override  protected String[] loadConfigAdminConfigurationFile() {      // String[0] = tell Camel the path of the .cfg file to use for OSGi ConfigAdmin in the blueprint XML file      // String[1] = tell Camel the persistence-id of the cm:property-placeholder in the blueprint XML file      return new String[]{"src/test/resources/etc/stuff.cfg", "stuff"};  } |

Notice that this method requires to return a String[] with 2 values. The 1st value is the path for the configuration file to load.  
The 2nd value is the persistence-id of the <cm:property-placeholder> tag.

The stuff.cfg file is just a plain properties file with the property placeholders such as:

|  |
| --- |
| ## this is a comment  greeting=Bye |

#### Using .cfg file and overriding properties for Blueprint property placeholders

You can do both as well. Here is a complete example. First we have the Blueprint XML file:

|  |
| --- |
| <blueprint xmlns="[http://www.osgi.org/xmlns/blueprint/v1.0.0"](http://www.osgi.org/xmlns/blueprint/v1.0.0%22)             xmlns:xsi="[http://www.w3.org/2001/XMLSchema-instance"](http://www.w3.org/2001/XMLSchema-instance%22)             xmlns:cm="[http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.1.0"](http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.1.0%22)             xsi:schemaLocation="  <http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.1.0> <http://aries.apache.org/schemas/blueprint-cm/blueprint-cm-1.1.0.xsd>  <http://www.osgi.org/xmlns/blueprint/v1.0.0> [http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd"](http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd%22)>      <!-- blueprint property placeholders, that will use etc/stuff.cfg as the properties file -->    <cm:property-placeholder persistent-id="stuff" update-strategy="reload">      <cm:default-properties>        <cm:property name="destination" value="to-be-replaced" />      </cm:default-properties>    </cm:property-placeholder>      <!-- a bean that uses a blueprint property placeholder -->    <bean id="myCoolBean" class="org.apache.camel.test.blueprint.MyCoolBean">      <property name="say" value="${greeting}"/>      <property name="echo" value="${echo}"/>    </bean>      <camelContext xmlns="[http://camel.apache.org/schema/blueprint"](http://camel.apache.org/schema/blueprint%22)>        <route>        <from uri="direct:start"/>        <bean ref="myCoolBean" method="saySomething"/>        <to uri="{{destination}}"/>        <bean ref="myCoolBean" method="echoSomething"/>        <to uri="{{destination}}"/>      </route>      </camelContext>    </blueprint> |

And in the unit test class we do as follows:

|  |
| --- |
| /\*\*   \* This example will load a Blueprint .cfdg file, and also override its property placeholders from this unit test   \* source code directly.   \*/  public class ConfigAdminLoadConfigurationFileAndOverrideTest extends CamelBlueprintTestSupport {        @Override      protected String getBlueprintDescriptor() {          // which blueprint XML file to use for this test          return "org/apache/camel/test/blueprint/configadmin-loadfileoverride.xml";      }        @Override      protected String[] loadConfigAdminConfigurationFile() {          // which .cfg file to use, and the name of the persistence-id          return new String[]{"src/test/resources/etc/stuff.cfg", "stuff"};      }        @Override      protected String useOverridePropertiesWithConfigAdmin(Dictionary props) throws Exception {          // override / add extra properties          props.put("destination", "mock:extra");            // return the persistence-id to use          return "stuff";      }        @Test      public void testConfigAdmin() throws Exception {          // regular unit test method          getMockEndpoint("mock:extra").expectedBodiesReceived("Bye World", "Yay Bye WorldYay Bye World");            template.sendBody("direct:start", "World");            assertMockEndpointsSatisfied();      }    } |

And the etc/stuff.cfg configuration file contains

|  |
| --- |
| greeting=Bye  echo=Yay  destination=mock:result |

### 

**Endpoints**

Camel supports the [Message Endpoint](http://camel.apache.org/message-endpoint.html) pattern using the [Endpoint](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/Endpoint.html) interface. Endpoints are usually created by a [Component](http://camel.apache.org/component.html) and Endpoints are usually referred to in the [DSL](http://camel.apache.org/dsl.html) via their [URIs](http://camel.apache.org/uris.html).

From an Endpoint you can use the following methods

* [createProducer()](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/Endpoint.html#createProducer%28%29) will create a [Producer](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/Producer.html) for sending message exchanges to the endpoint
* [createConsumer()](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/Endpoint.html#createConsumer%28org.apache.camel.Processor%29) implements the [Event Driven Consumer](http://camel.apache.org/event-driven-consumer.html) pattern for consuming message exchanges from the endpoint via a [Processor](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/Processor.html) when creating a [Consumer](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/Consumer.html)
* [createPollingConsumer()](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/Endpoint.html#createPollingConsumer%28%29) implements the [Polling Consumer](http://camel.apache.org/polling-consumer.html) pattern for consuming message exchanges from the endpoint via a [PollingConsumer](http://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/PollingConsumer.html)

## How do I configure endpoints?

There are a few different approaches to configuring components and endpoints.

### Using Java Code

You can explicitly configure a [Component](http://camel.apache.org/component.html) using Java code as shown in this [example](http://camel.apache.org/walk-through-an-example.html)

Or you can explicitly get hold of an [Endpoint](http://camel.apache.org/endpoint.html) and configure it using Java code as shown in the [Mock endpoint examples](http://camel.apache.org/mock.html).

|  |
| --- |
| SomeEndpoint endpoint = camelContext.getEndpoint("someURI", SomeEndpoint.class);  endpoint.setSomething("aValue"); |

### Using Guice

You can also use [Guice](http://camel.apache.org/guice.html) as the dependency injection framework. For example see the [Guice JMS Example](http://camel.apache.org/guice-jms-example.html)

### Using Spring XML

You can configure your [Component](http://camel.apache.org/component.html) or [Endpoint](http://camel.apache.org/endpoint.html) instances in your [Spring](http://camel.apache.org/spring.html) XML as follows.

|  |
| --- |
| <camelContext id="camel" xmlns="[http://camel.apache.org/schema/spring"](http://camel.apache.org/schema/spring%22)>      <jmxAgent id="agent" disabled="true"/>  </camelContext>    <bean id="activemq" class="org.apache.activemq.camel.component.ActiveMQComponent">    <property name="connectionFactory">      <bean class="org.apache.activemq.ActiveMQConnectionFactory">        <property name="brokerURL" value="<vm://localhost?>broker.persistent=false&amp;broker.useJmx=false"/>      </bean>    </property>  </bean> |

Which allows you to configure a component using some name (activemq in the above example), then you can refer to the component using **activemq:[queue:|topic:]destinationName**. This works by the SpringCamelContext lazily fetching components from the spring context for the scheme name you use for [Endpoint](http://camel.apache.org/endpoint.html) [URIs](http://camel.apache.org/uris.html)

### Using Endpoint URIs

Another approach is to use the URI syntax. The URI syntax supports the query notation. So for example with the [Mail](http://camel.apache.org/mail.html) component you can configure the password property via the URI

|  |
| --- |
| pop3://host:port?password=foo |

#### Referring beans from Endpoint URIs

**Available as of Camel 2.0**

When configuring endpoints using URI syntax you can now refer to beans in the [Registry](http://camel.apache.org/registry.html) using the # notation.  
If the parameter value starts with a # sign then Camel will lookup in the [Registry](http://camel.apache.org/registry.html) for a bean of the given type. For instance:

|  |
| --- |
| file://inbox?sorter=#mySpecialFileSorter |

Will lookup a bean with the id **mySpecialFileSorter** in the [Registry](http://camel.apache.org/registry.html).

#### Configuring parameter values using raw values, eg such as passwords

**Available as of Camel 2.11**

When configuring endpoint options using URI syntax, then the values is by default URI encoded. This can be a problem if you want to configure passwords and just use the value as is without any encoding. For example you may have a plus sign in the password, which would be decimal encoded by default.

So from Camel 2.11 onwards we made this easier as you can denote a parameter value to be **raw** using the following syntax RAW(value). eg the value starts with RAW( and then ends with the parenthesis ). Here is a little example:

|  |
| --- |
| .to("ftp:joe@myftpserver.com?password=RAW(se+re?t&23)&binary=true" |

In the above example, we have declare the password value as raw, and the actual password would be as typed, eg se+re?t&23.

#### Using property placeholders

Camel have extensive support for using property placeholders, which you can read more [about here](http://camel.apache.org/using-propertyplaceholder.html). For example in the ftp example above we can externalize the password to a .properties file.

For example configuring the property placeholder when using a [XML DSL](http://camel.apache.org/dsl.html), where we declare the location of the .properties file. Though we can also define this in Java code. See the [documentation](http://camel.apache.org/using-propertyplaceholder.html) for more details.

|  |
| --- |
| <camelContext ...>     <propertyPlaceholder id="properties" location="myftp.properties"/>     ...  </camelContext> |

And the Camel route now refers to the placeholder using the {{ key }} notation:

|  |
| --- |
| .to("ftp:joe@myftpserver.com?password={{myFtpPassword}}&binary=true" |

And have a myftp.properties file with password. Notice we still define the RAW(value) style to ensure the password is used as is

|  |
| --- |
| myFtpPassword=RAW(se+re?t&23) |

We could still have used the RAW(value) in the Camel route instead:

|  |
| --- |
| .to("ftp:joe@myftpserver.com?password=RAW({{myFtpPassword}})&binary=true" |

And then we would need to remove the RAW from the properties file:

|  |
| --- |
| myFtpPassword=se+re?t&23 |

To understand more about property placeholders, read the [documentation](http://camel.apache.org/using-propertyplaceholder.html).

### Configuring uris using endpoint with bean property style

**Available as of Camel 2.15**

Sometimes configuring endpoint uris may have many options, and therefore the uri can become long. In Java DSL you can break the uris into new lines as its just Java code, eg just concat the String. When using XML DSL then the uri is an attribute, eg <from uri="bla bla"/>. From Camel 2.15 onwards you can configure the endpoint separately, and from the routes refer to the endpoints using their shorthand ids.

|  |
| --- |
| <camelContext ...>      <endpoint id="foo" uri="ftp://foo@myserver">      <property name="password" value="secret"/>      <property name="recursive" value="true"/>      <property name="ftpClient.dataTimeout" value="30000"/>      <property name="ftpClient.serverLanguageCode" value="fr"/>    </endpoint>      <route>      <from uri="ref:foo"/>      ...    </route>  </camelContext> |

In the example above, the endpoint with id foo, is defined using <endpoint> which under the covers assembles this as an uri, with all the options, as if you have defined all the options directly in the uri. You can still configure some options in the uri, and then use <property> style for additional options, or to override options from the uri, such as:

|  |
| --- |
| <endpoint id="foo" uri="ftp://foo@myserver?recursive=true">    <property name="password" value="secret"/>    <property name="ftpClient.dataTimeout" value="30000"/>    <property name="ftpClient.serverLanguageCode" value="fr"/>  </endpoint> |

### Configuring long uris using new lines

**Available as of Camel 2.15**

Sometimes configuring endpoint uris may have many options, and therefore the uri can become long. In Java DSL you can break the uris into new lines as its just Java code, eg just concat the String. When using XML DSL then the uri is an attribute, eg <from uri="bla bla"/>. From Camel 2.15 onwards you can break the uri attribute using new line, such as shown below:

|  |
| --- |
| <route>    <from uri="<ftp://foo@myserver?>password=secret&amp;             recursive=true&amp;             ftpClient.dataTimeout=30000&amp;             ftpClientConfig.serverLanguageCode=fr"/>    <to uri="bean:doSomething"/>  </route> |

Notice that it still requires to use escape & as &ampl; in XML. Also you can have multiple options in one line, eg this is the same:

|  |
| --- |
| <route>    <from uri="<ftp://foo@myserver?>password=secret&amp;             recursive=true&amp;ftpClient.dataTimeout=30000&amp;             ftpClientConfig.serverLanguageCode=fr"/>    <to uri="bean:doSomething"/>  </route> |

# Web Console

**Deprecated**

The web console module from Apache Camel distribution is deprecated and will be removed from Camel 2.16 onwards.  
Instead we encourage users to look at some of the alternative console projects, which some are listed in the bottom of this page.

The Camel Web Console is available from versions 2.0 onwards and provides a full access over a RESTful API to camel endpoints, messages and routes.

Camel 2.5 or better

The [Web Console](http://camel.apache.org/web-console.html) from Camel 2.5 onwards requires JDK 1.6 as minimum to run.

## Download and Run the Console

Download the console from Maven Central at <http://repo2.maven.org/maven2/org/apache/camel/camel-web-standalone/>

Then from the command line type

|  |
| --- |
| java -jar camel-web-standalone-2.8.0.jar |

You should now be able to point your browser at: <http://localhost:8080/>

## Build and Run the Console

First get the latest [Source](http://camel.apache.org/source.html) then from the command line type

|  |
| --- |
| cd components/camel-web  mvn jetty:run |

Then point your web browse at <http://localhost:8080/>

You should be able to do things like

* browse the available endpoints
* browse the messages on an endpoint if it is a [BrowsableEndpoint](http://camel.apache.org/browsableendpoint.html)
* send a message to an endpoint
* create new endpoints

### Running using Tomcat

**Available as of Camel 2.10**

You can also run the web console from an embedded Apache Tomcat using

|  |
| --- |
| cd components/camel-web  mvn tomcat7:run |

Then point your web browse at <http://localhost:8080/>

## Embedding web console in your own web application.

See these relevant discussions for more information (more information to come soon):

<http://camel.465427.n5.nabble.com/Embedded-web-console-td478885.html>

<http://camel.465427.n5.nabble.com/example-app-to-embed-camel-web-console-td4512075.html>

## REST API

Camel comes with a full RESTful API for interacting with the Camel context, the available endpoints and routes. You can browse details of the running API via <http://localhost:8080/api>

The web application uses mostly the same URIs for the HTML representation of a resource (e.g. **/endpoints**) as the JSON and XML representations. To help rendering the different representations in your browser you can append **.xml**, **.html**, **.json** or even **.dot** to URLs.

For example viewing these URLs are equivalent

|  |  |
| --- | --- |
| **URL** | **Same as** |
| <http://localhost:8080/endpoints.xml> | <http://localhost:8080/endpoints>  with Accept header of text/xml or application/xml |
| <http://localhost:8080/endpoints.json> | <http://localhost:8080/endpoints>  with Accept header of application/json |
| <http://localhost:8080/routes.dot> | <http://localhost:8080/routes>  with Accept header of text/vnd.graphviz |

For more details try viewing the [API documentation in your local Camel instance](http://localhost:8080/api)

## Route Viewing and Editing through Web Console

Web Console provide route viewing and editing functionality. You can view your route via <http://localhost:8080/routes/yourRouteId> and it default present the route in XML.  
Camel uses JAXB to process the XML route definitions.

# External Web Consoles

There are a number of external web consoles for Apache Camel in separate open source projects:

|  |  |  |
| --- | --- | --- |
| **External Project** | **Description** | |
| [hawtio](http://hawt.io/) | hawtio is an open source HTML5 web application for visualising, managing, tracing and debugging Camel routes & endpoints, ActiveMQ brokers, JMX, OSGi and logging. | |
| [CamelWatch](http://sksamuel.github.com/camelwatch/) | A web app for monitoring Camel applications. | |
| [RHQ](http://www.jboss.org/rhq) | RHQ is an open source operational monitoring tool which has support for Apache Camel (along with other Apache projects like Tomcat, httpd, ActiveMQ etc) | |
| You addressed the issue of "dynamic URIs for Consumers" back in July-2009:  <http://camel.465427.n5.nabble.com/Dynamic-consumers-with-Camel-td476433.html>  However, all those use-cases, except for maybe #4, web console, which  I can't run all are about  calculating the URI \*before\* the context is started. What I'm looking  for is a way to create and insert  or update a polling consumer's URI - at runtime - \*after\* the context  is started.  For example:  from("direct:start")  .beanRef("config")  .convertBodyTo(Document.class)  .setHeader("Ftp\_URI").xquery(  "concat('ftp://'" +...some more xpaths into the  inbound config doc...), String.class)  .pollEnrich(/\* whoops! this is referenced at route  setup time, not route runtime... - it won't work \*/)  .to("log:...");  So the question, more generally, is is it possible to alter the route  definition - at runtime? i.e. after the route/context are started?  You sorted of hinted at this in that July-2009 posting when you said:  "2) You can always stop, modify and start a route in Camel at runtime."  I don't see how that would work since I would likely be attempting to  modify the route at runtime  from a bean method in the route - or are you suggesting one route  stops/modifies another route?  So the way I see it is - I would create the ftp route with a bogus URI  and option "&startScheduler=false",  on the ftp endpoint, or autoStart=false on the route. Then in  \*another\* route - the "configurer" route,  I guess, access the ftp route and reconfigure the endpoint with the  real settings, then start it's poll scheduler (or start it's route).  Something like that? Any easier, more direct way?  Thanks,  Chris  [route](http://qnalist.com/tags/route/) [at](http://qnalist.com/tags/at/) [ftp](http://qnalist.com/tags/ftp/) [for](http://qnalist.com/tags/for/) [this](http://qnalist.com/tags/this/)  [**Reply To : It Is Possible With Camel To Use Dynamic Consumer? \*\*after Context Start\*\***](mailto:users@camel.apache.org?In-Reply-To=%3CCAK=acOLitEXxX5admuvYpScPojmRX7iDQzOh1pkPT7fB7nFsVQ@mail.gmail.com%3E&Subject=It%20is%20possible%20with%20Camel%20to%20use%20dynamic%20consumer?%20**after%20context%20start**)  asked **Mar 31 2013 at 20:08**  Chris Wolf's gravatar image  [Chris Wolf](http://qnalist.com/user/chris-wolf/) | |

1 Replies for : It Is Possible With Camel To Use Dynamic Consumer? \*\*after Context Start\*\*

|  |  |
| --- | --- |
|  | You may want to see a little about consumer template here  <http://camel.apache.org/polling-consumer.html>  Notice the the consumer template polls one message at a time.  If you want to "download all available files" you would need to run  the code in a while loop as the example on that link.  Though you can also add a new route at runtime with the from uri  computed. And then stop and remove the route if not longer needed. |