Red Hat Integration 2020-Q2

Getting Started with Camel Kafka Connector

TECHNOLOGY PREVIEW - Using Camel components as Kafka connectors

Last Updated: 2020-09-25

Red Hat Integration 2020-Q2 Getting Started with Camel Kafka Connector

TECHNOLOGY PREVIEW - Using Camel components as Kafka connectors

Legal Notice

Abstract

This guide introduces Camel Kafka Connector, explains how to install into AMQ Streams and Kafka Connect on OpenShift, and how to get started with example Camel Kafka connectors. This guide also provides reference details on the Camel Kafka connectors that you can configure in this release.

Table of Contents

CHAPTER 1. INTRODUCTION TO CAMEL KAFKA CONNECTOR

This chapter introduces the features, concepts, and distributions provided by Camel Kafka Connector:

- Section 1.1, “Camel Kafka Connector overview”
- Section 1.2, “Camel Kafka Connector features”
- Section 1.3, “Camel Kafka Connector architecture”
- Section 1.4, “Camel Kafka Connector distributions”

IMPORTANT

Camel Kafka Connector is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production.

These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

1.1. CAMEL KAFKA CONNECTOR OVERVIEW

Apache Camel is a highly flexible open source integration framework for connecting a wide range of different systems, which is based on standard Enterprise Integration Patterns (EIPs). Apache Kafka Connect is the Kafka-native approach for connecting to external systems, which is specifically designed for event-driven architectures.

Camel Kafka Connector enables you to use standard Camel components as Kafka Connect connectors. This widens the scope of possible integrations beyond the external systems supported by Kafka Connect connectors alone. Camel Kafka Connector works as an adapter that makes the popular Camel component ecosystem available in Kafka-based AMQ Streams on OpenShift.

Camel Kafka Connector provides a user-friendly way to configure Camel components directly in the Kafka Connect framework. Using Camel Kafka Connector, you can leverage Camel components for integration with different systems by connecting to or from Camel Kafka sink or source connectors. You do not need to write any code, and can include the appropriate connector JARs in your Kafka Connect image and configure connector options using custom resources.

Camel Kafka Connector is built on Apache Camel Kafka Connector, which is a subproject of the Apache Camel open source community. Camel Kafka Connector is fully integrated with OpenShift Container Platform, AMQ Streams, and Kafka Connect.

Camel Kafka Connector is available with the Red Hat Integration - Camel K distribution for cloud-native integration on OpenShift. Camel K is a lightweight integration framework built from Apache Camel K that runs natively in the cloud on OpenShift. Camel K is specifically designed for serverless and microservice architectures.

Additional resources

- Apache Camel Kafka Connector GitHub project

- Apache Camel Kafka Connector website
- Deploying Red Hat Integration - Camel K on OpenShift

1.2. CAMEL KAFKA CONNECTOR FEATURES

The Camel Kafka Connector Technology Preview includes the following main features:

- OpenShift Container Platform 4.4 or 4.3
- AMQ Streams 1.5
- Kafka Connect 2.5
- Camel 3.1
- Selected Camel Kafka connectors

Table 1.1. Camel Kafka connectors in Technology Preview

Connector	Sink/source
Amazon AWS Kinesis	Sink and source
Amazon AWS S3	Sink and source
Cassandra Query Language (CQL)	Sink only
Elasticsearch	Sink only
Java Message Service (JMS)	Sink and source
Salesforce	Source only
Syslog	Source only

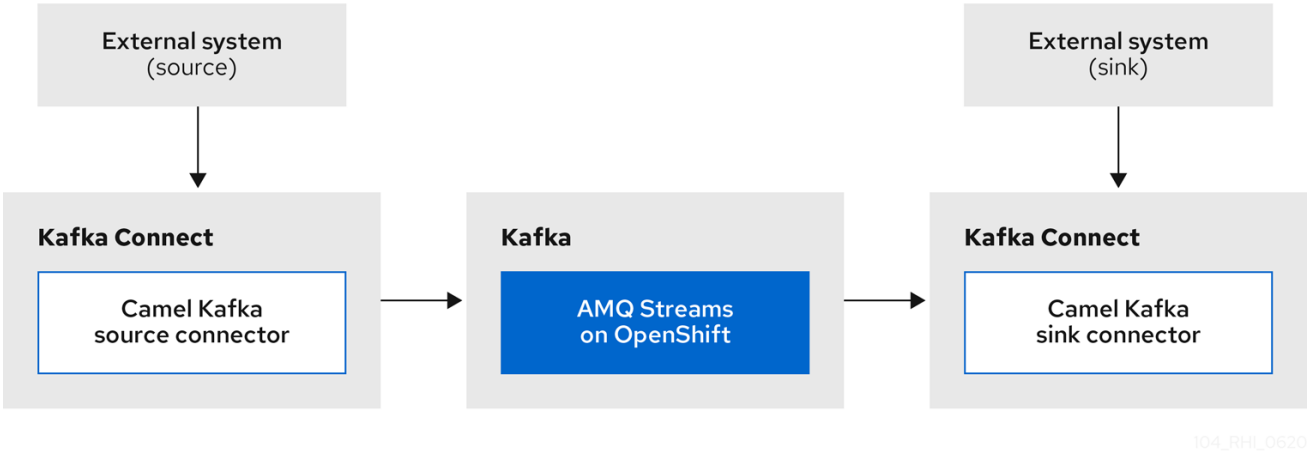
1.3. CAMEL KAFKA CONNECTOR ARCHITECTURE

AMQ Streams is a distributed and scalable streaming platform based on Apache Kafka that includes a publish/subscribe messaging broker. Kafka Connect provides a framework to integrate Kafka-based systems with external systems. Using Kafka Connect, you can configure *source* and *sink* connectors to stream data from external systems into and out of a Kafka broker.

Camel Kafka Connector reuses the flexibility of Camel components and makes them available in Kafka Connect as source and sink connectors that you can use to stream data into and out of AMQ Streams. For example, you can ingest data from Amazon Web Services for processing using an AWS S3 source connector, or consolidate events stored in Kafka into an Elasticsearch instance for analytics using an Elasticsearch sink connector.

The following diagram shows a simplified view of the Camel Kafka Connector cloud-native integration architecture based on AMQ Streams on OpenShift:

Figure 1.1. Camel Kafka Connector architecture



Kafka Connect concepts

The main Kafka Connect concepts include:

Source connector

Source connectors work like consumers and pull data from external systems into Kafka topics to make the data available for stream processing. For example, these external source systems include Amazon Web Services or Java Message Service.

Sink connector

Sink connectors work like producers and push data from Kafka topics into external systems for offline analysis. For example, these external sink systems include Cassandra, Syslog, or Elasticsearch.

Sink/source task

Tasks are typically created by a sink or source connector and are responsible for handling the data.

Key/value converter

Key/value converters can serialize/deserialize the key or value of a Kafka message in various formats.

Transformer

Transformers can manipulate Kafka message content.

Camel Kafka Connector configuration

You can use Camel Kafka Connector configuration to specify the following:

- Kafka Connect configuration options
- Camel route definitions
- Camel configuration options

Additional resources

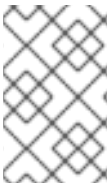
- Apache Kafka Connect user documentation

1.4. CAMEL KAFKA CONNECTOR DISTRIBUTIONS

The Camel Kafka Connector distributions are bundled with Red Hat Integration - Camel K:

Table 1.2. Camel Kafka Connector available distributions

Distribution	Description	Location
Maven repository	Maven artifacts for Camel Kafka Connector	Software Downloads > Red Hat Integration - Camel K
Source code	Source code for Camel Kafka Connector	Software Downloads > Red Hat Integration - Camel K
Demonstration examples	Camel Kafka Connector examples and Debezium and Apache Camel Kafka Connector community example	• AWS S3 source to JMS sink connectors • CQL sink connector • CQL source connector • Debezium PostgreSQL connector (Debezium community)



NOTE

You must have a subscription for Red Hat Integration and be logged into the Red Hat Customer Portal to access the Camel Kafka Connector distributions available with Red Hat Integration - Camel K.

CHAPTER 2. DEPLOYING CAMEL KAFKA CONNECTOR WITH AMQ STREAMS

This chapter explains how to install Camel Kafka Connector into AMQ Streams on OpenShift and how to get started with example connectors.

2.1. CONFIGURING AUTHENTICATION WITH REGISTRY.REDHAT.IO

You must configure authentication with the **registry.redhat.io** container registry before you can use AMQ Streams and Kafka Connect Soure-2-Image (S2I) to deploy Camel Kafka Connector on OpenShift.

Prerequisites

- You must have cluster administrator access to an OpenShift Container Platform cluster.
- You must have the OpenShift **oc** client tool installed. For more details, see the OpenShift CLI documentation.

Procedure

1. Log into your OpenShift cluster as administrator, for example:

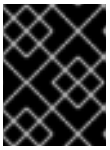
```
$ oc login --user system:admin --token=my-token --server=https://my-cluster.example.com:6443
```

2. Open the project in which you wish to deploy Camel Kafka Connector, for example:

```
$ oc project myproject
```

3. Create a **docker-registry** secret using your Red Hat Customer Portal account, and replace **PULL_SECRET_NAME** with the name of the secret that you wish to create:

```
$ oc create secret docker-registry PULL_SECRET_NAME \
  --docker-server=registry.redhat.io \
  --docker-username=CUSTOMER_PORTAL_USERNAME \
  --docker-password=CUSTOMER_PORTAL_PASSWORD \
  --docker-email=EMAIL_ADDRESS
```



IMPORTANT

You must create the **docker-registry** secret in every OpenShift project namespace that will include the image streams and use **registry.redhat.io**.

4. Link the secret to your service account to use the secret for pulling images. The following example uses the **default** service account:

```
$ oc secrets link default PULL_SECRET_NAME --for=pull
```

The service account name must match the name that the service account Pod uses.

5. Link the secret to the **builder** service account in the namespace in which you plan to use Kafka Connect S2I:

```
$ oc secrets link builder PULL_SECRET_NAME
```

NOTE

If you do not want to use your Red Hat account username and password to create the pull secret, you should create an authentication token by using a registry service account.

Additional resources

- Red Hat Registry authentication
- Red Hat Registry service accounts

2.2. INSTALLING AMQ STREAMS AND KAFKA CONNECT S2I ON OPENSHIFT

AMQ Streams and Kafka Connect with Source-2-Image (S2I) are required to install Camel Kafka Connector. If you do not already have AMQ Streams installed, you can install the AMQ Streams Operator on your OpenShift cluster from the OperatorHub. The OperatorHub is available from the OpenShift Container Platform web console and provides an interface for cluster administrators to discover and install Operators. For more details, see the OpenShift documentation.

Prerequisites

- You must have cluster administrator access to an OpenShift Container Platform cluster.
- You must have authenticated with **registry.redhat.io** using the steps in Section 2.1, "Configuring authentication with registry.redhat.io".
- See Using AMQ Streams on OpenShift for detailed information on installing AMQ Streams and Kafka Connect S2I. This section shows a simple default example of installing using the OpenShift OperatorHub.

Procedure

1. In the OpenShift Container Platform web console, log in using an account with cluster administrator privileges.
2. Select your project from the **Project** drop-down in the toolbar, for example, **myproject**. This must be the project in which you have authenticated with **registry.redhat.io**.
3. In the left navigation menu, click **Operators** > **OperatorHub**.

4. In the **Filter by keyword** text box, enter **AMQ** to find the **Red Hat Integration - AMQ Streams** Operator.

5. Read the information about the Operator, and click **Install**. This displays the **Create Operator Subscription** page.

6. Select your subscription settings:

   - **Installation Mode** > **A specific namespace on the cluster**> **myproject**
   - **Update Channel** > **stable**
   - **Approval Strategy** > **Automatic**

NOTE

These settings depend on the specific requirements of your environment. For more details, see OpenShift documentation on Adding Operators to a cluster.

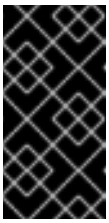
7. Click **Subscribe**. This displays the **Operators** > **Installed Operators** page.

8. Wait a few moments until the **Status** for the AMQ Streams Operator displays **Succeeded** and the subscription is **Up to Date**.

9. Create a new Kafka broker cluster:

   a. Under **Red Hat Integration - AMQ Streams**> **Provided APIs** > **Kafka**, click **Create Instance** to create a new Kafka broker cluster.

   b. Edit the custom resource definition as appropriate, and click **Create**.



IMPORTANT

The default example creates a Kafka cluster with 3 Zookeeper nodes and 3 Kafka nodes with **ephemeral** storage. This temporary storage is suitable for development and testing only, and not for a production environment. For more details, see Using AMQ Streams on OpenShift .

10. Create a new Kafka Connect S2I cluster:

    a. Under **Red Hat Integration - AMQ Streams**> **Provided APIs** > **Kafka Connect S2I**, click **Create Instance** to create a new Kafka Connect cluster with OpenShift Source-2-Image support.

    b. Edit the custom resource definition as appropriate, and click **Create**. For more details on using Kafka Connect with S2I, see Using AMQ Streams on OpenShift .

11. Select **Workloads** > **Pods** to verify that the deployed resources are running on OpenShift.

Additional resources

- OpenShift documentation on Adding Operators to a cluster

2.3. DEPLOYING CAMEL KAFKA CONNECTOR IN AMQ STREAMS ON OPENSHIFT

This section explains how to use Kafka Connect Source-2-Image (S2I) to add your Camel Kafka connectors to your existing Docker-based Kafka Connect image to build a new image. This section also shows how to create an instance of a Camel Kafka connector plug-in using an example AWS S3 Camel Kafka connector.

Prerequisites

- You must have cluster administrator access to an OpenShift Container Platform cluster.
- You must have the OpenShift **oc** client tool installed. For more details, see the OpenShift CLI documentation.
- You must have installed AMQ Streams and Kafka Connect with S2I support on your OpenShift cluster. For more details, see Section 2.2, "Installing AMQ Streams and Kafka Connect S2I on OpenShift".
- You must have downloaded the connectors available for Camel Kafka Connector from Software Downloads > Red Hat Integration - Camel K.

Procedure

1. Log into your OpenShift cluster as administrator, for example:

```
$ oc login --user system:admin --token=my-token --server=https://my-cluster.example.com:6443
```

2. Change to the project in which Kafka Connect S2I is installed:

```
$ oc project myproject
```

3. Add your downloaded connectors to the existing Kafka Connect Docker image build, and then wait for the new image build to finish and be configured with the new connectors. For example:

```
$ oc start-build my-connect-cluster-connect --from-dir=./camel-kafka-connector/connectors/ --follow
Uploading directory "camel-kafka-connector/connectors" as binary input for the build ...
...
Uploading finished
build.build.openshift.io/my-connect-cluster-connect-2 started
Receiving source from STDIN as archive ...
Caching blobs under "/var/cache/blobs".
Getting image source signatures
Copying blob sha256:5ed7b62ff462957d0ee8956db7a787d8e17c1bdee7a78c57c917298019f77ea2
...
Writing manifest to image destination
Storing signatures
Generating dockerfile with builder image image-registry.openshift-image-registry.svc:5000/myproject/my-connect-cluster-connect-source@sha256:12d5ed92510941f1569faa449665e9fc6ea544e67b7ae189ec6b8df434e121f4
STEP 1: FROM image-registry.openshift-image-registry.svc:5000/myproject/my-connect-
```

```
cluster-connect-
source@sha256:12d5ed92510941f1569faa449665e9fc6ea544e67b7ae189ec6b8df434e121f
4
STEP 2: LABEL "io.openshift.build.image"="image-registry.openshift-image-
registry.svc:5000/myproject/my-connect-cluster-connect-
source@sha256:12d5ed92510941f1569faa449665e9fc6ea544e67b7ae189ec6b8df434e121f4"
"io.openshift.build.source-location"="/tmp/build/inputs"
STEP 3: ENV OPENSHIFT_BUILD_NAME="my-connect-cluster-connect-2"
OPENSHIFT_BUILD_NAMESPACE="myproject"
STEP 4: USER root
STEP 5: COPY upload/src /tmp/src
STEP 6: RUN chown -R 1001:0 /tmp/src
STEP 7: USER 1001
STEP 8: RUN /opt/kafka/s2i/assemble
Assembling plugins into custom plugin directory /tmp/kafka-plugins
Moving plugins to /tmp/kafka-plugins
STEP 9: CMD /opt/kafka/s2i/run
STEP 10: COMMIT temp.builder.openshift.io/myproject/my-connect-cluster-connect-
2:d0873588
Getting image source signatures
Copying blob
sha256:edf3aa290fb3c255a84fe836109093fbfeef65c08544f655fad8d6afb53868ba
...
Writing manifest to image destination
Storing signatures
0d392e3df3edc0801f0b7091ba99e2a666008531ccb5271cd0d4b54901dac0b9
0d392e3df3edc0801f0b7091ba99e2a666008531ccb5271cd0d4b54901dac0b9

Pushing image image-registry.openshift-image-registry.svc:5000/myproject/my-connect-
cluster-connect:latest ...
Getting image source signatures
Copying blob
sha256:06ea991a3b933c49058585f82006648f8702a33b5de8725e2fe85724f18a2ff4
...
Writing manifest to image destination
Storing signatures
Successfully pushed image-registry.openshift-image-registry.svc:5000/myproject/my-
connect-cluster-
connect@sha256:9db57d33df6d0494ea6ee6e4696fcaf79eb81aabeb0bbc180dec5324d33e7ed
a
Push successful
```

4. Check that the Camel Kafka connectors are available in your Kafka Connect cluster as follows:

```
$ oc exec -i -c kafka my-cluster-kafka-0 -- curl -s http://my-connect-cluster-connect-
api:8083/connector-plugins
```

You should see something like the following output:

```
[{"class":"org.apache.camel.kafkaconnector.CamelSinkConnector","type":"sink","version":"0.0
.1-SNAPSHOT"},
{"class":"org.apache.camel.kafkaconnector.CamelSourceConnector","type":"source","version
":"0.0.1-SNAPSHOT"},
{"class":"org.apache.kafka.connect.file.FileStreamSinkConnector","type":"sink","version":"2.3.
```

```
0"},
{"class":"org.apache.kafka.connect.file.FileStreamSourceConnector","type":"source","version"
:"2.3.0"}]
```

5. Use the following annotation to enable instantiating Camel Kafka connectors using a specific custom resource:

```
$ oc annotate kafkaconnects2is my-connect-cluster strimzi.io/use-connector-resources=true
```

IMPORTANT

When the **use-connector-resources** option is enabled, do not use the Kafka Connect API server. The Kafka Connect Operator will revert any changes that you make.

6. Create the connector instance by creating a specific custom resource that includes your connector configuration. The following example shows the configuration for an AWS S3 connector plug-in:

```
$ oc apply -f - << EOF
apiVersion: kafka.strimzi.io/v1alpha1
kind: KafkaConnector
metadata:
  name: s3-source-connector
  namespace: myproject
  labels:
    strimzi.io/cluster: my-connect-cluster
spec:
  class: org.apache.camel.kafkaconnector.awss3.CamelAwss3SourceConnector
  tasksMax: 1
  config:
    key.converter: org.apache.kafka.connect.storage.StringConverter
    value.converter: org.apache.camel.kafkaconnector.awss3.converters.S3ObjectConverter
    topics: s3-topic
    camel.source.path.bucketNameOrArn: camel-connector-test
    camel.source.endpoint.autocloseBody: false
    camel.source.maxPollDuration: 10000
    camel.component.aws-s3.accessKey: xxx
    camel.component.aws-s3.secretKey: xxx
    camel.component.aws-s3.region: xxx
EOF

kafkaconnector.kafka.strimzi.io/s3-source-connector created
```

7. Check the status of your connector using the following command, for example:

```
$ oc get kctr --selector strimzi.io/cluster=my-connect-cluster -o yaml
```

8. You can also run the Kafka console consumer to see the messages received from the topic:

```
$ oc exec -i -c kafka my-cluster-kafka-0 -- bin/kafka-console-consumer.sh --bootstrap-server
localhost:9092 --topic s3-topic --from-beginning
```

Additional resources

- Apache Camel Kafka Connector installation instructions on OpenShift
- Using AMQ Streams on OpenShift

2.4. DEPLOYING CAMEL KAFKA CONNECTOR EXAMPLES

This section describes how to deploy the following Camel Kafka Connector examples:

- AWS S3 source to JMS sink connectors
- CQL sink connector
- Debezium PostgreSQL connector (community example)

Prerequisites

- See the *What is needed* section in the readmes shown in the Procedure section.

Procedure

- Perform the steps described in the GitHub readme for the each demonstration example:
  - AWS S3 source connector to JMS sink connector
  - CQL sink connector
  - Debezium PostgreSQL connector and Apache Camel Kafka Connector

Additional resources

- Using the Apache Camel Kafka Connectors with Strimzi

CHAPTER 3. CAMEL KAFKA CONNECTOR DEVELOPER REFERENCE

This chapter provides reference information on the Camel Kafka connectors that you can configure using Camel Kafka Connector.

IMPORTANT

This Technology Preview release includes a targeted subset of the available Apache Camel Kafka connectors. Additional connectors will be added to Camel Kafka Connector in future releases.

Table 3.1. Camel Kafka Connector configuration

Connector	Sink	Source
Amazon AWS Kinesis	Camel AWS Kinesis sink connector	Camel AWS Kinesis source connector
Amazon AWS S3	Camel AWS s3 sink connector	Camel AWS s3 source connector
Cassandra Query Language	Camel Cassandra CQL sink connector	-
Elasticsearch	Camel Elasticsearch sink connector	-
Java Message Service	Camel JMS sink connector	Camel JMS source connector
Salesforce	-	Camel Salesforce source connector
Syslog	-	Camel Syslog source connector

3.1. CAMEL-AWS-KINESIS-KAFKA-CONNECTOR SINK CONFIGURATION

When using camel-aws-kinesis-kafka-connector as sink make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-aws-kinesis-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-aws-kinesis sink connector supports 21 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.streamName	Name of the stream	null	HIGH
camel.sink.endpoint.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	MEDIUM
camel.sink.endpoint.proxyHost	To define a proxy host when instantiating the Kinesis client	null	MEDIUM
camel.sink.endpoint.proxyPort	To define a proxy port when instantiating the Kinesis client	null	MEDIUM
camel.sink.endpoint.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.sink.endpoint.region	The region in which Kinesis client needs to work. When using this parameter, the configuration will expect the capitalized name of the region (for example AP_EAST_1)You'll need to use the name Regions.EU_WEST_1.name()	null	MEDIUM
camel.sink.endpoint.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.sink.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.sink.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM
camel.sink.endpoint.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.sink.endpoint.secretKey	Amazon AWS Secret Key	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws-kinesis.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	MEDIUM
camel.component.aws-kinesis.configuration	The component configuration	null	MEDIUM
camel.component.aws-kinesis.proxyHost	To define a proxy host when instantiating the Kinesis client	null	MEDIUM
camel.component.aws-kinesis.proxyPort	To define a proxy port when instantiating the Kinesis client	null	MEDIUM
camel.component.aws-kinesis.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.component.aws-kinesis.region	The region in which Kinesis client needs to work. When using this parameter, the configuration will expect the capitalized name of the region (for example AP_EAST_1)You'll need to use the name Regions.EU_WEST_1.name()	null	MEDIUM
camel.component.aws-kinesis.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.component.aws-kinesis.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.aws-kinesis.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.component.aws-kinesis.secretKey	Amazon AWS Secret Key	null	MEDIUM

3.2. CAMEL-AWS-KINESIS-KAFKA-CONNECTOR SOURCE CONFIGURATION

When using camel-aws-kinesis-kafka-connector as source make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-aws-kinesis-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-aws-kinesis source connector supports 49 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.streamName	Name of the stream	null	HIGH
camel.source.endpoint.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	MEDIUM
camel.source.endpoint.proxyHost	To define a proxy host when instantiating the Kinesis client	null	MEDIUM
camel.source.endpoint.proxyPort	To define a proxy port when instantiating the Kinesis client	null	MEDIUM
camel.source.endpoint.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.source.endpoint.region	The region in which Kinesis client needs to work. When using this parameter, the configuration will expect the capitalized name of the region (for example AP_EAST_1)You'll need to use the name Regions.EU_WEST_1.name()	null	MEDIUM
camel.source.endpoint.bridgeErrorHandler	Allows for bridging the consumer to the Camel routing Error Handler, which mean any exceptions occurred while the consumer is trying to pickup incoming messages, or the likes, will now be processed as a message and handled by the routing Error Handler. By default the consumer will use the org.apache.camel.spi.ExceptionHandler to deal with exceptions, that will be logged at WARN or ERROR level and ignored.	false	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.iteratorType	Defines where in the Kinesis stream to start getting records One of: [AT_SEQUENCE_NUMBER] [AFTER_SEQUENCE_NUMBER] [TRIM_HORIZON] [LATEST] [AT_TIMESTAMP]	"TRIM_HORIZON"	MEDIUM
camel.source.endpoint.maxResultsPerRequest	Maximum number of records that will be fetched in each poll	1	MEDIUM
camel.source.endpoint.sendEmptyMessageWhenIdle	If the polling consumer did not poll any files, you can enable this option to send an empty message (no body) instead.	false	MEDIUM
camel.source.endpoint.sequenceNumber	The sequence number to start polling from. Required if iteratorType is set to AFTER_SEQUENCE_NUMBER or AT_SEQUENCE_NUMBER	null	MEDIUM
camel.source.endpoint.shardClosed	Define what will be the behavior in case of shard closed. Possible value are ignore, silent and fail. In case of ignore a message will be logged and the consumer will restart from the beginning,in case of silent there will be no logging and the consumer will start from the beginning,in case of fail a ReachedClosedStateException will be raised One of: [ignore] [fail] [silent]	"ignore"	MEDIUM
camel.source.endpoint.shardId	Defines which shardId in the Kinesis stream to get records from	null	MEDIUM
camel.source.endpoint.exceptionHandler	To let the consumer use a custom ExceptionHandler. Notice if the option bridgeErrorHandler is enabled then this option is not in use. By default the consumer will deal with exceptions, that will be logged at WARN or ERROR level and ignored.	null	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.pollStrategy	A pluggable org.apache.camel.PollingConsumerPollingStrategy allowing you to provide your custom implementation to control error handling usually occurred during the poll operation before an Exchange have been created and being routed in Camel.	null	MEDIUM
camel.source.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.source.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM
camel.source.endpoint.backoffErrorThreshold	The number of subsequent error polls (failed due some error) that should happen before the backoffMultipler should kick-in.	null	MEDIUM
camel.source.endpoint.backoffIdleThreshold	The number of subsequent idle polls that should happen before the backoffMultipler should kick-in.	null	MEDIUM
camel.source.endpoint.backoffMultiplier	To let the scheduled polling consumer backoff if there has been a number of subsequent idles/errors in a row. The multiplier is then the number of polls that will be skipped before the next actual attempt is happening again. When this option is in use then backoffIdleThreshold and/or backoffErrorThreshold must also be configured.	null	MEDIUM
camel.source.endpoint.delay	Milliseconds before the next poll.	500L	MEDIUM
camel.source.endpoint.greedy	If greedy is enabled, then the ScheduledPollConsumer will run immediately again, if the previous run polled 1 or more messages.	false	MEDIUM
camel.source.endpoint.initialDelay	Milliseconds before the first poll starts.	1000L	MEDIUM
camel.source.endpoint.repeatCount	Specifies a maximum limit of number of fires. So if you set it to 1, the scheduler will only fire once. If you set it to 5, it will only fire five times. A value of zero or negative means fire forever.	0L	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.runLoggingLevel	The consumer logs a start/complete log line when it polls. This option allows you to configure the logging level for that. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"TRACE"	MEDIUM
camel.source.endpoint.scheduledExecutorService	Allows for configuring a custom/shared thread pool to use for the consumer. By default each consumer has its own single threaded thread pool.	null	MEDIUM
camel.source.endpoint.scheduler	To use a cron scheduler from either camel-spring or camel-quartz component One of: [none] [spring] [quartz]	"none"	MEDIUM
camel.source.endpoint.schedulerProperties	To configure additional properties when using a custom scheduler or any of the Quartz, Spring based scheduler.	null	MEDIUM
camel.source.endpoint.startScheduler	Whether the scheduler should be auto started.	true	MEDIUM
camel.source.endpoint.timeUnit	Time unit for initialDelay and delay options. One of: [NANOSECONDS] [MICROSECONDS] [MILLISECONDS] [SECONDS] [MINUTES] [HOURS] [DAYS]	"MILLISECONDS"	MEDIUM
camel.source.endpoint.useFixedDelay	Controls if fixed delay or fixed rate is used. See ScheduledExecutorService in JDK for details.	true	MEDIUM
camel.source.endpoint.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.source.endpoint.secretKey	Amazon AWS Secret Key	null	MEDIUM
camel.component.aws-kinesis.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	MEDIUM
camel.component.aws-kinesis.configuration	The component configuration	null	MEDIUM
camel.component.aws-kinesis.proxyHost	To define a proxy host when instantiating the Kinesis client	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws-kinesis.proxyPort	To define a proxy port when instantiating the Kinesis client	null	MEDIUM
camel.component.aws-kinesis.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.component.aws-kinesis.region	The region in which Kinesis client needs to work. When using this parameter, the configuration will expect the capitalized name of the region (for example AP_EAST_1)You'll need to use the name Regions.EU_WEST_1.name()	null	MEDIUM
camel.component.aws-kinesis.bridgeErrorHandler	Allows for bridging the consumer to the Camel routing Error Handler, which mean any exceptions occurred while the consumer is trying to pickup incoming messages, or the likes, will now be processed as a message and handled by the routing Error Handler. By default the consumer will use the org.apache.camel.spi.ExceptionHandler to deal with exceptions, that will be logged at WARN or ERROR level and ignored.	false	MEDIUM
camel.component.aws-kinesis.iteratorType	Defines where in the Kinesis stream to start getting records One of: [AT_SEQUENCE_NUMBER] [AFTER_SEQUENCE_NUMBER] [TRIM_HORIZON] [LATEST] [AT_TIMESTAMP]	"TRIM_HORIZON"	MEDIUM
camel.component.aws-kinesis.maxResultsPerRequest	Maximum number of records that will be fetched in each poll	1	MEDIUM
camel.component.aws-kinesis.sequenceNumber	The sequence number to start polling from. Required if iteratorType is set to AFTER_SEQUENCE_NUMBER or AT_SEQUENCE_NUMBER	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws-kinesis.shardClosed	Define what will be the behavior in case of shard closed. Possible value are ignore, silent and fail. In case of ignore a message will be logged and the consumer will restart from the beginning,in case of silent there will be no logging and the consumer will start from the beginning,in case of fail a ReachedClosedStateException will be raised One of: [ignore] [fail] [silent]	"ignore"	MEDIUM
camel.component.aws-kinesis.shardId	Defines which shardId in the Kinesis stream to get records from	null	MEDIUM
camel.component.aws-kinesis.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.aws-kinesis.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.component.aws-kinesis.secretKey	Amazon AWS Secret Key	null	MEDIUM

3.3. CAMEL-AWS-S3-KAFKA-CONNECTOR SINK CONFIGURATION

When using camel-aws-s3-kafka-connector as sink make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-aws-s3-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-aws-s3 sink connector supports 63 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.bucketNameOrArn	Bucket name or ARN	null	HIGH
camel.sink.endpoint.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	MEDIUM
camel.sink.endpoint.autoCreateBucket	Setting the autocreation of the bucket	true	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.endpointConfiguration	Amazon AWS Endpoint Configuration	null	MEDIUM
camel.sink.endpoint.pathStyleAccess	Whether or not the S3 client should use path style access	false	MEDIUM
camel.sink.endpoint.policy	The policy for this queue to set in the com.amazonaws.services.s3.AmazonS3#setBucketPolicy() method.	null	MEDIUM
camel.sink.endpoint.proxyHost	To define a proxy host when instantiating the S3 client	null	MEDIUM
camel.sink.endpoint.proxyPort	Specify a proxy port to be used inside the client definition.	null	MEDIUM
camel.sink.endpoint.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.sink.endpoint.region	The region in which S3 client needs to work. When using this parameter, the configuration will expect the capitalized name of the region (for example AP_EAST_1) You'll need to use the name Regions.EU_WEST_1.name()	null	MEDIUM
camel.sink.endpoint.useIAMCredentials	Set whether the S3 client should expect to load credentials on an EC2 instance or to expect static credentials to be passed in.	false	MEDIUM
camel.sink.endpoint.encryptionMaterials	The encryption materials to use in case of Symmetric/Asymmetric client usage	null	MEDIUM
camel.sink.endpoint.useEncryption	Define if encryption must be used or not	false	MEDIUM
camel.sink.endpoint.deleteAfterWrite	Delete file object after the S3 file has been uploaded	false	MEDIUM
camel.sink.endpoint.keyName	Setting the key name for an element in the bucket through endpoint parameter	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.sink.endpoint.multiPartUpload	If it is true, camel will upload the file with multi part format, the part size is decided by the option of partSize	false	MEDIUM
camel.sink.endpoint.operation	The operation to do in case the user don't want to do only an upload One of: [copyObject] [deleteBucket] [listBuckets] [downloadLink]	null	MEDIUM
camel.sink.endpoint.partSize	Setup the partSize which is used in multi part upload, the default size is 25M.	26214400L	MEDIUM
camel.sink.endpoint.serverSideEncryption	Sets the server-side encryption algorithm when encrypting the object using AWS-managed keys. For example use AES256.	null	MEDIUM
camel.sink.endpoint.storageClass	The storage class to set in the com.amazonaws.services.s3.model.PutObjectRequest request.	null	MEDIUM
camel.sink.endpoint.awsKMSKeyId	Define the id of KMS key to use in case KMS is enabled	null	MEDIUM
camel.sink.endpoint.useAwsKMS	Define if KMS must be used or not	false	MEDIUM
camel.sink.endpoint.accelerateModeEnabled	Define if Accelerate Mode enabled is true or false	false	MEDIUM
camel.sink.endpoint.chunkedEncodingDisabled	Define if disabled Chunked Encoding is true or false	false	MEDIUM
camel.sink.endpoint.dualstackEnabled	Define if Dualstack enabled is true or false	false	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.forceGlobalBucketAccessEnabled	Define if Force Global Bucket Access enabled is true or false	false	MEDIUM
camel.sink.endpoint.payloadSigningEnabled	Define if Payload Signing enabled is true or false	false	MEDIUM
camel.sink.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.sink.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM
camel.sink.endpoint.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.sink.endpoint.secretKey	Amazon AWS Secret Key	null	MEDIUM
camel.component.aws-s3.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	MEDIUM
camel.component.aws-s3.autoCreateBucket	Setting the autocreation of the bucket	true	MEDIUM
camel.component.aws-s3.configuration	The component configuration	null	MEDIUM
camel.component.aws-s3.endpointConfiguration	Amazon AWS Endpoint Configuration	null	MEDIUM
camel.component.aws-s3.pathStyleAccess	Whether or not the S3 client should use path style access	false	MEDIUM
camel.component.aws-s3.policy	The policy for this queue to set in the com.amazonaws.services.s3.AmazonS3#setBucketPolicy() method.	null	MEDIUM
camel.component.aws-s3.proxyHost	To define a proxy host when instantiating the S3 client	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws-s3.proxyPort	Specify a proxy port to be used inside the client definition.	null	MEDIUM
camel.component.aws-s3.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.component.aws-s3.region	The region in which S3 client needs to work. When using this parameter, the configuration will expect the capitalized name of the region (for example AP_EAST_1) You'll need to use the name Regions.EU_WEST_1.name()	null	MEDIUM
camel.component.aws-s3.useIAMCredentials	Set whether the S3 client should expect to load credentials on an EC2 instance or to expect static credentials to be passed in.	false	MEDIUM
camel.component.aws-s3.encryptionMaterials	The encryption materials to use in case of Symmetric/Asymmetric client usage	null	MEDIUM
camel.component.aws-s3.useEncryption	Define if encryption must be used or not	false	MEDIUM
camel.component.aws-s3.deleteAfterWrite	Delete file object after the S3 file has been uploaded	false	MEDIUM
camel.component.aws-s3.keyName	Setting the key name for an element in the bucket through endpoint parameter	null	MEDIUM
camel.component.aws-s3.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.component.aws-s3.multiPartUpload	If it is true, camel will upload the file with multi part format, the part size is decided by the option of partSize	false	MEDIUM

Name	Description	Default	Priority
camel.component.aws-s3.operation	The operation to do in case the user don't want to do only an upload One of: [copyObject] [deleteBucket] [listBuckets] [downloadLink]	null	MEDIUM
camel.component.aws-s3.partSize	Setup the partSize which is used in multi part upload, the default size is 25M.	26214400L	MEDIUM
camel.component.aws-s3.serverSideEncryption	Sets the server-side encryption algorithm when encrypting the object using AWS-managed keys. For example use AES256.	null	MEDIUM
camel.component.aws-s3.storageClass	The storage class to set in the com.amazonaws.services.s3.model.PutObjectRequest request.	null	MEDIUM
camel.component.aws-s3.awsKMSKeyId	Define the id of KMS key to use in case KMS is enabled	null	MEDIUM
camel.component.aws-s3.useAwsKMS	Define if KMS must be used or not	false	MEDIUM
camel.component.aws-s3.accelerateModeEnabled	Define if Accelerate Mode enabled is true or false	false	MEDIUM
camel.component.aws-s3.chunkedEncodingDisabled	Define if disabled Chunked Encoding is true or false	false	MEDIUM
camel.component.aws-s3.dualstackEnabled	Define if Dualstack enabled is true or false	false	MEDIUM
camel.component.aws-s3.forceGlobalBucketAccess Enabled	Define if Force Global Bucket Access enabled is true or false	false	MEDIUM
camel.component.aws-s3.payloadSigningEnabled	Define if Payload Signing enabled is true or false	false	MEDIUM
camel.component.aws-s3.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.aws-s3.accessKey	Amazon AWS Access Key	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws-s3.secretKey	Amazon AWS Secret Key	null	MEDIUM

3.4. CAMEL-AWS-S3-KAFKA-CONNECTOR SOURCE CONFIGURATION

When using camel-aws-s3-kafka-connector as source make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-aws-s3-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-aws-s3 source connector supports 77 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.bucketNameOrArn	Bucket name or ARN	null	HIGH
camel.source.endpoint.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	MEDIUM
camel.source.endpoint.autoCreateBucket	Setting the autocreation of the bucket	true	MEDIUM
camel.source.endpoint.endpointConfiguration	Amazon AWS Endpoint Configuration	null	MEDIUM
camel.source.endpoint.pathStyleAccess	Whether or not the S3 client should use path style access	false	MEDIUM
camel.source.endpoint.policy	The policy for this queue to set in the com.amazonaws.services.s3.AmazonS3#setBucketPolicy() method.	null	MEDIUM
camel.source.endpoint.proxyHost	To define a proxy host when instantiating the S3 client	null	MEDIUM
camel.source.endpoint.proxyPort	Specify a proxy port to be used inside the client definition.	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.source.endpoint.region	The region in which S3 client needs to work. When using this parameter, the configuration will expect the capitalized name of the region (for example AP_EAST_1) You'll need to use the name Regions.EU_WEST_1.name()	null	MEDIUM
camel.source.endpoint.useIAMCredentials	Set whether the S3 client should expect to load credentials on an EC2 instance or to expect static credentials to be passed in.	false	MEDIUM
camel.source.endpoint.encryptionMaterials	The encryption materials to use in case of Symmetric/Asymmetric client usage	null	MEDIUM
camel.source.endpoint.useEncryption	Define if encryption must be used or not	false	MEDIUM
camel.source.endpoint.bridgeErrorHandler	Allows for bridging the consumer to the Camel routing Error Handler, which mean any exceptions occurred while the consumer is trying to pickup incoming messages, or the likes, will now be processed as a message and handled by the routing Error Handler. By default the consumer will use the org.apache.camel.spi.ExceptionHandler to deal with exceptions, that will be logged at WARN or ERROR level and ignored.	false	MEDIUM
camel.source.endpoint.deleteAfterRead	Delete objects from S3 after they have been retrieved. The delete is only performed if the Exchange is committed. If a rollback occurs, the object is not deleted. If this option is false, then the same objects will be retrieve over and over again on the polls. Therefore you need to use the Idempotent Consumer EIP in the route to filter out duplicates. You can filter using the S3Constants#BUCKET_NAME and S3Constants#KEY headers, or only the S3Constants#KEY header.	true	MEDIUM
camel.source.endpoint.delimiter	The delimiter which is used in the com.amazonaws.services.s3.model.ListObjects Request to only consume objects we are interested in.	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.fileName	To get the object from the bucket with the given file name	null	MEDIUM
camel.source.endpoint.includeBody	If it is true, the exchange body will be set to a stream to the contents of the file. If false, the headers will be set with the S3 object metadata, but the body will be null. This option is strongly related to autocloseBody option. In case of setting includeBody to true and autocloseBody to false, it will be up to the caller to close the S3Object stream. Setting autocloseBody to true, will close the S3Object stream automatically.	true	MEDIUM
camel.source.endpoint.maxConnections	Set the maxConnections parameter in the S3 client configuration	60	MEDIUM
camel.source.endpoint.maxMessagesPerPoll	Gets the maximum number of messages as a limit to poll at each polling. Gets the maximum number of messages as a limit to poll at each polling. The default value is 10. Use 0 or a negative number to set it as unlimited.	10	MEDIUM
camel.source.endpoint.prefix	The prefix which is used in the com.amazonaws.services.s3.model.ListObjects Request to only consume objects we are interested in.	null	MEDIUM
camel.source.endpoint.sendEmptyMessageWhenIdle	If the polling consumer did not poll any files, you can enable this option to send an empty message (no body) instead.	false	MEDIUM
camel.source.endpoint.autocloseBody	If this option is true and includeBody is true, then the S3Object.close() method will be called on exchange completion. This option is strongly related to includeBody option. In case of setting includeBody to true and autocloseBody to false, it will be up to the caller to close the S3Object stream. Setting autocloseBody to true, will close the S3Object stream automatically.	true	MEDIUM
camel.source.endpoint.exceptionHandler	To let the consumer use a custom ExceptionHandler. Notice if the option bridgeErrorHandler is enabled then this option is not in use. By default the consumer will deal with exceptions, that will be logged at WARN or ERROR level and ignored.	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	MEDIUM
camel.source.endpoint.pollStrategy	A pluggable org.apache.camel.PollingConsumerPollingStrategy allowing you to provide your custom implementation to control error handling usually occurred during the poll operation before an Exchange have been created and being routed in Camel.	null	MEDIUM
camel.source.endpoint.accelerateModeEnabled	Define if Accelerate Mode enabled is true or false	false	MEDIUM
camel.source.endpoint.chunkedEncodingDisabled	Define if disabled Chunked Encoding is true or false	false	MEDIUM
camel.source.endpoint.dualstackEnabled	Define if Dualstack enabled is true or false	false	MEDIUM
camel.source.endpoint.forceGlobalBucketAccess Enabled	Define if Force Global Bucket Access enabled is true or false	false	MEDIUM
camel.source.endpoint.payloadSigningEnabled	Define if Payload Signing enabled is true or false	false	MEDIUM
camel.source.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.source.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM
camel.source.endpoint.backoffErrorThreshold	The number of subsequent error polls (failed due some error) that should happen before the backoffMultipler should kick-in.	null	MEDIUM
camel.source.endpoint.backoffIdleThreshold	The number of subsequent idle polls that should happen before the backoffMultipler should kick-in.	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.backoffMultiplier	To let the scheduled polling consumer backoff if there has been a number of subsequent idles/errors in a row. The multiplier is then the number of polls that will be skipped before the next actual attempt is happening again. When this option is in use then backoffIdleThreshold and/or backoffErrorThreshold must also be configured.	null	MEDIUM
camel.source.endpoint.delay	Milliseconds before the next poll.	500L	MEDIUM
camel.source.endpoint.greedy	If greedy is enabled, then the ScheduledPollConsumer will run immediately again, if the previous run polled 1 or more messages.	false	MEDIUM
camel.source.endpoint.initialDelay	Milliseconds before the first poll starts.	1000L	MEDIUM
camel.source.endpoint.repeatCount	Specifies a maximum limit of number of fires. So if you set it to 1, the scheduler will only fire once. If you set it to 5, it will only fire five times. A value of zero or negative means fire forever.	0L	MEDIUM
camel.source.endpoint.runLoggingLevel	The consumer logs a start/complete log line when it polls. This option allows you to configure the logging level for that. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"TRACE"	MEDIUM
camel.source.endpoint.scheduledExecutorService	Allows for configuring a custom/shared thread pool to use for the consumer. By default each consumer has its own single threaded thread pool.	null	MEDIUM
camel.source.endpoint.scheduler	To use a cron scheduler from either camel-spring or camel-quartz component One of: [none] [spring] [quartz]	"none"	MEDIUM
camel.source.endpoint.schedulerProperties	To configure additional properties when using a custom scheduler or any of the Quartz, Spring based scheduler.	null	MEDIUM
camel.source.endpoint.startScheduler	Whether the scheduler should be auto started.	true	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.timeUnit	Time unit for initialDelay and delay options. One of: [NANOSECONDS] [MICROSECONDS] [MILLISECONDS] [SECONDS] [MINUTES] [HOURS] [DAYS]	"MILLISECONDS"	MEDIUM
camel.source.endpoint.useFixedDelay	Controls if fixed delay or fixed rate is used. See ScheduledExecutorService in JDK for details.	true	MEDIUM
camel.source.endpoint.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.source.endpoint.secretKey	Amazon AWS Secret Key	null	MEDIUM
camel.component.aws-s3.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	MEDIUM
camel.component.aws-s3.autoCreateBucket	Setting the autocreation of the bucket	true	MEDIUM
camel.component.aws-s3.configuration	The component configuration	null	MEDIUM
camel.component.aws-s3.endpointConfiguration	Amazon AWS Endpoint Configuration	null	MEDIUM
camel.component.aws-s3.pathStyleAccess	Whether or not the S3 client should use path style access	false	MEDIUM
camel.component.aws-s3.policy	The policy for this queue to set in the com.amazonaws.services.s3.AmazonS3#setBucketPolicy() method.	null	MEDIUM
camel.component.aws-s3.proxyHost	To define a proxy host when instantiating the S3 client	null	MEDIUM
camel.component.aws-s3.proxyPort	Specify a proxy port to be used inside the client definition.	null	MEDIUM
camel.component.aws-s3.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM

Name	Description	Default	Priority
camel.component.aws-s3.region	The region in which S3 client needs to work. When using this parameter, the configuration will expect the capitalized name of the region (for example AP_EAST_1) You'll need to use the name Regions.EU_WEST_1.name()	null	MEDIUM
camel.component.aws-s3.useIAMCredentials	Set whether the S3 client should expect to load credentials on an EC2 instance or to expect static credentials to be passed in.	false	MEDIUM
camel.component.aws-s3.encryptionMaterials	The encryption materials to use in case of Symmetric/Asymmetric client usage	null	MEDIUM
camel.component.aws-s3.useEncryption	Define if encryption must be used or not	false	MEDIUM
camel.component.aws-s3.bridgeErrorHandler	Allows for bridging the consumer to the Camel routing Error Handler, which mean any exceptions occurred while the consumer is trying to pickup incoming messages, or the likes, will now be processed as a message and handled by the routing Error Handler. By default the consumer will use the org.apache.camel.spi.ExceptionHandler to deal with exceptions, that will be logged at WARN or ERROR level and ignored.	false	MEDIUM
camel.component.aws-s3.deleteAfterRead	Delete objects from S3 after they have been retrieved. The delete is only performed if the Exchange is committed. If a rollback occurs, the object is not deleted. If this option is false, then the same objects will be retrieve over and over again on the polls. Therefore you need to use the Idempotent Consumer EIP in the route to filter out duplicates. You can filter using the S3Constants#BUCKET_NAME and S3Constants#KEY headers, or only the S3Constants#KEY header.	true	MEDIUM
camel.component.aws-s3.delimiter	The delimiter which is used in the com.amazonaws.services.s3.model.ListObjects Request to only consume objects we are interested in.	null	MEDIUM
camel.component.aws-s3.fileName	To get the object from the bucket with the given file name	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws-s3.includeBody	If it is true, the exchange body will be set to a stream to the contents of the file. If false, the headers will be set with the S3 object metadata, but the body will be null. This option is strongly related to autocloseBody option. In case of setting includeBody to true and autocloseBody to false, it will be up to the caller to close the S3Object stream. Setting autocloseBody to true, will close the S3Object stream automatically.	true	MEDIUM
camel.component.aws-s3.prefix	The prefix which is used in the com.amazonaws.services.s3.model.ListObjects Request to only consume objects we are interested in.	null	MEDIUM
camel.component.aws-s3.autocloseBody	If this option is true and includeBody is true, then the S3Object.close() method will be called on exchange completion. This option is strongly related to includeBody option. In case of setting includeBody to true and autocloseBody to false, it will be up to the caller to close the S3Object stream. Setting autocloseBody to true, will close the S3Object stream automatically.	true	MEDIUM
camel.component.aws-s3.accelerateModeEnabled	Define if Accelerate Mode enabled is true or false	false	MEDIUM
camel.component.aws-s3.chunkedEncodingDisabled	Define if disabled Chunked Encoding is true or false	false	MEDIUM
camel.component.aws-s3.dualstackEnabled	Define if Dualstack enabled is true or false	false	MEDIUM
camel.component.aws-s3.forceGlobalBucketAccess Enabled	Define if Force Global Bucket Access enabled is true or false	false	MEDIUM
camel.component.aws-s3.payloadSigningEnabled	Define if Payload Signing enabled is true or false	false	MEDIUM

Name	Description	Default	Priority
camel.component.aws-s3.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.aws-s3.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.component.aws-s3.secretKey	Amazon AWS Secret Key	null	MEDIUM

Examples

Here is an example of configuration of the source connector

```
name=CamelAWSS3SourceConnector
connector.class=org.apache.camel.kafkaconnector.awss3.CamelAwss3SourceConnector
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.camel.kafkaconnector.awss3.converters.S3ObjectConverter

camel.source.maxPollDuration=10000

topics=mytopic

camel.source.url=aws-s3://camel-kafka-connector?autocloseBody=false

camel.component.aws-s3.access-key=xxxx
camel.component.aws-s3.secret-key=yyyy
camel.component.aws-s3.region=EU_WEST_1
```

In this example we are polling the bucket camel-kafka-connector as source.

3.5. CAMEL-CQL-KAFKA-CONNECTOR SINK CONFIGURATION

When using camel-cql-kafka-connector as sink make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-cql-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-cql sink connector supports 19 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.beanRef	beanRef is defined using bean:id	null	MEDIUM

Name	Description	Default	Priority
camel.sink.path.hosts	Hostname(s) cassansdra server(s). Multiple hosts can be separated by comma.	null	MEDIUM
camel.sink.path.port	Port number of cassansdra server(s)	null	MEDIUM
camel.sink.path.keyspace	Keyspace to use	null	MEDIUM
camel.sink.endpoint.cluster	To use the Cluster instance (you would normally not use this option)	null	MEDIUM
camel.sink.endpoint.clusterName	Cluster name	null	MEDIUM
camel.sink.endpoint.consistencyLevel	Consistency level to use One of: [ANY] [ONE] [TWO] [THREE] [QUORUM] [ALL] [LOCAL_QUORUM] [EACH_QUORUM] [SERIAL] [LOCAL_SERIAL] [LOCAL_ONE]	null	MEDIUM
camel.sink.endpoint.cql	CQL query to perform. Can be overridden with the message header with key CamelCqlQuery.	null	MEDIUM
camel.sink.endpoint.loadBalancingPolicy	To use a specific LoadBalancingPolicy	null	MEDIUM
camel.sink.endpoint.password	Password for session authentication	null	MEDIUM
camel.sink.endpoint.prepareStatements	Whether to use PreparedStatements or regular Statements	true	MEDIUM
camel.sink.endpoint.resultSetConversionStrategy	To use a custom class that implements logic for converting ResultSet into message body ALL, ONE, LIMIT_10, LIMIT_100...	null	MEDIUM
camel.sink.endpoint.session	To use the Session instance (you would normally not use this option)	null	MEDIUM
camel.sink.endpoint.username	Username for session authentication	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.sink.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.sink.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM
camel.component.cql.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.component.cql.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM

3.6. CAMEL-ELASTICSEARCH-REST-KAFKA-CONNECTOR SINK CONFIGURATION

When using camel-elasticsearch-rest-kafka-connector as sink make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
```

```
    <artifactId>camel-elasticsearch-rest-kafka-connector</artifactId>
    <version>x.x.x</version>
    <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-elasticsearch-rest sink connector supports 33 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.clusterName	Name of the cluster	null	HIGH
camel.sink.endpoint.connectionTimeout	The time in ms to wait before connection will timeout.	30000	MEDIUM
camel.sink.endpoint.disconnect	Disconnect after it finish calling the producer	false	MEDIUM
camel.sink.endpoint.enableSniffer	Enable automatically discover nodes from a running Elasticsearch cluster	false	MEDIUM
camel.sink.endpoint.enableSSL	Enable SSL	false	MEDIUM
camel.sink.endpoint.from	Starting index of the response.	null	MEDIUM
camel.sink.endpoint.hostAddresses	Comma separated list with ip:port formatted remote transport addresses to use.	null	HIGH
camel.sink.endpoint.indexName	The name of the index to act against	null	MEDIUM
camel.sink.endpoint.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.sink.endpoint.maxRetryTimeout	The time in ms before retry	30000	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.operation	What operation to perform One of: [Index] [Update] [Bulk] [BulkIndex] [GetById] [MultiGet] [MultiSearch] [Delete] [DeleteIndex] [Search] [Exists] [Ping]	null	MEDIUM
camel.sink.endpoint.scrollKeepAliveMs	Time in ms during which elasticsearch will keep search context alive	60000	MEDIUM
camel.sink.endpoint.size	Size of the response.	null	MEDIUM
camel.sink.endpoint.sniffAfterFailureDelay	The delay of a sniff execution scheduled after a failure (in milliseconds)	60000	MEDIUM
camel.sink.endpoint.snifferInterval	The interval between consecutive ordinary sniff executions in milliseconds. Will be honoured when sniffOnFailure is disabled or when there are no failures between consecutive sniff executions	300000	MEDIUM
camel.sink.endpoint.socketTimeout	The timeout in ms to wait before the socket will timeout.	30000	MEDIUM
camel.sink.endpoint.useScroll	Enable scroll usage	false	MEDIUM
camel.sink.endpoint.waitForActiveShards	Index creation waits for the write consistency number of shards to be available	1	MEDIUM
camel.sink.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.sink.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM

Name	Description	Default	Priority
camel.component.elasticsearch-rest.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.component.elasticsearch-rest.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.elasticsearch-rest.client	To use an existing configured Elasticsearch client, instead of creating a client per endpoint. This allow to customize the client with specific settings.	null	MEDIUM
camel.component.elasticsearch-rest.connectionTimeout	The time in ms to wait before connection will timeout.	30000	MEDIUM
camel.component.elasticsearch-rest.enableSniffer	Enable automatically discover nodes from a running Elasticsearch cluster	"false"	MEDIUM
camel.component.elasticsearch-rest.hostAddresses	Comma separated list with ip:port formatted remote transport addresses to use. The ip and port options must be left blank for hostAddresses to be considered instead.	null	MEDIUM
camel.component.elasticsearch-rest.maxRetryTimeout	The time in ms before retry	30000	MEDIUM
camel.component.elasticsearch-rest.sniffAfterFailureDelay	The delay of a sniff execution scheduled after a failure (in milliseconds)	60000	MEDIUM

Name	Description	Default	Priority
camel.component.elasticsearch-rest.snifferInterval	The interval between consecutive ordinary sniff executions in milliseconds. Will be honoured when sniffOnFailure is disabled or when there are no failures between consecutive sniff executions	300000	MEDIUM
camel.component.elasticsearch-rest.socketTimeout	The timeout in ms to wait before the socket will timeout.	30000	MEDIUM
camel.component.elasticsearch-rest.enableSSL	Enable SSL	"false"	MEDIUM
camel.component.elasticsearch-rest.password	Password for authenticate	null	MEDIUM
camel.component.elasticsearch-rest.user	Basic authenticate user	null	MEDIUM

3.7. CAMEL-JMS-KAFKA-CONNECTOR SINK CONFIGURATION

When using camel-jms-kafka-connector as sink make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-jms-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-jms sink connector supports 141 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.destinationType	The kind of destination to use One of: [queue] [topic] [temp-queue] [temp-topic]	"queue"	MEDIUM
camel.sink.path.destinationName	Name of the queue or topic to use as destination	null	HIGH
camel.sink.endpoint.clientId	Sets the JMS client ID to use. Note that this value, if specified, must be unique and can only be used by a single JMS connection instance. It is typically only required for durable topic subscriptions. If using Apache ActiveMQ you may prefer to use Virtual Topics instead.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.connectionFactory	The connection factory to be use. A connection factory must be configured either on the component or endpoint.	null	MEDIUM
camel.sink.endpoint.disableReplyTo	Specifies whether Camel ignores the JMSReplyTo header in messages. If true, Camel does not send a reply back to the destination specified in the JMSReplyTo header. You can use this option if you want Camel to consume from a route and you do not want Camel to automatically send back a reply message because another component in your code handles the reply message. You can also use this option if you want to use Camel as a proxy between different message brokers and you want to route message from one system to another.	false	MEDIUM
camel.sink.endpoint.durableSubscriptionName	The durable subscriber name for specifying durable topic subscriptions. The clientId option must be configured as well.	null	MEDIUM
camel.sink.endpoint.jmsMessageType	Allows you to force the use of a specific javax.jms.Message implementation for sending JMS messages. Possible values are: Bytes, Map, Object, Stream, Text. By default, Camel would determine which JMS message type to use from the In body type. This option allows you to specify it. One of: [Bytes] [Map] [Object] [Stream] [Text]	null	MEDIUM
camel.sink.endpoint.testConnectionOnStartup	Specifies whether to test the connection on startup. This ensures that when Camel starts that all the JMS consumers have a valid connection to the JMS broker. If a connection cannot be granted then Camel throws an exception on startup. This ensures that Camel is not started with failed connections. The JMS producers is tested as well.	false	MEDIUM
camel.sink.endpoint.deliveryDelay	Sets delivery delay to use for send calls for JMS. This option requires JMS 2.0 compliant broker.	-1L	MEDIUM
camel.sink.endpoint.deliveryMode	Specifies the delivery mode to be used. Possibles values are those defined by javax.jms.DeliveryMode. NON_PERSISTENT = 1 and PERSISTENT = 2. One of: [1] [2]	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.deliveryPersistent	Specifies whether persistent delivery is used by default.	true	MEDIUM
camel.sink.endpoint.explicitQosEnabled	Set if the deliveryMode, priority or timeToLive qualities of service should be used when sending messages. This option is based on Spring's JmsTemplate. The deliveryMode, priority and timeToLive options are applied to the current endpoint. This contrasts with the preserveMessageQos option, which operates at message granularity, reading QoS properties exclusively from the Camel In message headers.	"false"	MEDIUM
camel.sink.endpoint.formatDateHeadersToIso8601	Sets whether JMS date properties should be formatted according to the ISO 8601 standard.	false	MEDIUM
camel.sink.endpoint.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.sink.endpoint.preserveMessageQos	Set to true, if you want to send message using the QoS settings specified on the message, instead of the QoS settings on the JMS endpoint. The following three headers are considered JMSPriority, JMSDeliveryMode, and JMSExpiration. You can provide all or only some of them. If not provided, Camel will fall back to use the values from the endpoint instead. So, when using this option, the headers override the values from the endpoint. The explicitQosEnabled option, by contrast, will only use options set on the endpoint, and not values from the message header.	false	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.priority	Values greater than 1 specify the message priority when sending (where 0 is the lowest priority and 9 is the highest). The explicitQosEnabled option must also be enabled in order for this option to have any effect. One of: [1] [2] [3] [4] [5] [6] [7] [8] [9]	4	MEDIUM
camel.sink.endpoint.replyToConcurrentConsumers	Specifies the default number of concurrent consumers when doing request/reply over JMS. See also the maxMessagesPerTask option to control dynamic scaling up/down of threads.	1	MEDIUM
camel.sink.endpoint.replyToMaxConcurrentConsumers	Specifies the maximum number of concurrent consumers when using request/reply over JMS. See also the maxMessagesPerTask option to control dynamic scaling up/down of threads.	null	MEDIUM
camel.sink.endpoint.replyToOnTimeoutMaxConcurrent Consumers	Specifies the maximum number of concurrent consumers for continue routing when timeout occurred when using request/reply over JMS.	1	MEDIUM
camel.sink.endpoint.replyToOverride	Provides an explicit ReplyTo destination in the JMS message, which overrides the setting of replyTo. It is useful if you want to forward the message to a remote Queue and receive the reply message from the ReplyTo destination.	null	MEDIUM
camel.sink.endpoint.replyToType	Allows for explicitly specifying which kind of strategy to use for replyTo queues when doing request/reply over JMS. Possible values are: Temporary, Shared, or Exclusive. By default Camel will use temporary queues. However if replyTo has been configured, then Shared is used by default. This option allows you to use exclusive queues instead of shared ones. See Camel JMS documentation for more details, and especially the notes about the implications if running in a clustered environment, and the fact that Shared reply queues has lower performance than its alternatives Temporary and Exclusive. One of: [Temporary] [Shared] [Exclusive]	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.requestTimeout	The timeout for waiting for a reply when using the InOut Exchange Pattern (in milliseconds). The default is 20 seconds. You can include the header CamelJmsRequestTimeout to override this endpoint configured timeout value, and thus have per message individual timeout values. See also the requestTimeoutCheckerInterval option.	20000L	MEDIUM
camel.sink.endpoint.timeToLive	When sending messages, specifies the time-to-live of the message (in milliseconds).	-1L	MEDIUM
camel.sink.endpoint.allowAdditionalHeaders	This option is used to allow additional headers which may have values that are invalid according to JMS specification. For example some message systems such as WMQ do this with header names using prefix JMS_IBM_MQMD_ containing values with byte array or other invalid types. You can specify multiple header names separated by comma, and use as suffix for wildcard matching.	null	MEDIUM
camel.sink.endpoint.allowNullBody	Whether to allow sending messages with no body. If this option is false and the message body is null, then an JMSException is thrown.	true	MEDIUM
camel.sink.endpoint.alwaysCopyMessage	If true, Camel will always make a JMS message copy of the message when it is passed to the producer for sending. Copying the message is needed in some situations, such as when a replyToDestinationSelectorName is set (incidentally, Camel will set the alwaysCopyMessage option to true, if a replyToDestinationSelectorName is set)	false	MEDIUM
camel.sink.endpoint.correlationProperty	When using InOut exchange pattern use this JMS property instead of JMSCorrelationID JMS property to correlate messages. If set messages will be correlated solely on the value of this property JMSCorrelationID property will be ignored and not set by Camel.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.disableTimeToLive	Use this option to force disabling time to live. For example when you do request/reply over JMS, then Camel will by default use the requestTimeout value as time to live on the message being sent. The problem is that the sender and receiver systems have to have their clocks synchronized, so they are in sync. This is not always so easy to archive. So you can use disableTimeToLive=true to not set a time to live value on the sent message. Then the message will not expire on the receiver system. See below in section About time to live for more details.	false	MEDIUM
camel.sink.endpoint.forceSendOriginalMessage	When using mapJmsMessage=false Camel will create a new JMS message to send to a new JMS destination if you touch the headers (get or set) during the route. Set this option to true to force Camel to send the original JMS message that was received.	false	MEDIUM
camel.sink.endpoint.includeSentJMSMessageID	Only applicable when sending to JMS destination using InOnly (eg fire and forget). Enabling this option will enrich the Camel Exchange with the actual JMSMessageID that was used by the JMS client when the message was sent to the JMS destination.	false	MEDIUM
camel.sink.endpoint.replyToCacheLevelName	Sets the cache level by name for the reply consumer when doing request/reply over JMS. This option only applies when using fixed reply queues (not temporary). Camel will by default use: CACHE_CONSUMER for exclusive or shared w/ replyToSelectorName. And CACHE_SESSION for shared without replyToSelectorName. Some JMS brokers such as IBM WebSphere may require to set the replyToCacheLevelName=CACHE_NONE to work. Note: If using temporary queues then CACHE_NONE is not allowed, and you must use a higher value such as CACHE_CONSUMER or CACHE_SESSION. One of: [CACHE_AUTO] [CACHE_CONNECTION] [CACHE_CONSUMER] [CACHE_NONE] [CACHE_SESSION]	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.replyToDestinationSelectorName	Sets the JMS Selector using the fixed name to be used so you can filter out your own replies from the others when using a shared queue (that is, if you are not using a temporary reply queue).	null	MEDIUM
camel.sink.endpoint.streamMessageTypeEnabled	Sets whether StreamMessage type is enabled or not. Message payloads of streaming kind such as files, InputStream, etc will either by sent as BytesMessage or StreamMessage. This option controls which kind will be used. By default BytesMessage is used which enforces the entire message payload to be read into memory. By enabling this option the message payload is read into memory in chunks and each chunk is then written to the StreamMessage until no more data.	false	MEDIUM
camel.sink.endpoint.allowSerializedHeaders	Controls whether or not to include serialized headers. Applies only when transferExchange is true. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level.	false	MEDIUM
camel.sink.endpoint.artemisStreamingEnabled	Whether optimizing for Apache Artemis streaming mode.	true	MEDIUM
camel.sink.endpoint.asyncStartListener	Whether to startup the JmsConsumer message listener asynchronously, when starting a route. For example if a JmsConsumer cannot get a connection to a remote JMS broker, then it may block while retrying and/or failover. This will cause Camel to block while starting routes. By setting this option to true, you will let routes startup, while the JmsConsumer connects to the JMS broker using a dedicated thread in asynchronous mode. If this option is used, then beware that if the connection could not be established, then an exception is logged at WARN level, and the consumer will not be able to receive messages; You can then restart the route to retry.	false	MEDIUM
camel.sink.endpoint.asyncStopListener	Whether to stop the JmsConsumer message listener asynchronously, when stopping a route.	false	MEDIUM
camel.sink.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.destinationResolver	A pluggable org.springframework.jms.support.destination.DestinationResolver that allows you to use your own resolver (for example, to lookup the real destination in a JNDI registry).	null	MEDIUM
camel.sink.endpoint.errorHandler	Specifies a org.springframework.util.ErrorHandler to be invoked in case of any uncaught exceptions thrown while processing a Message. By default these exceptions will be logged at the WARN level, if no errorHandler has been configured. You can configure logging level and whether stack traces should be logged using errorHandlerLoggingLevel and errorHandlerLogStackTrace options. This makes it much easier to configure, than having to code a custom errorHandler.	null	MEDIUM
camel.sink.endpoint.exceptionListener	Specifies the JMS Exception Listener that is to be notified of any underlying JMS exceptions.	null	MEDIUM
camel.sink.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to filter header to and from Camel message.	null	MEDIUM
camel.sink.endpoint.idleConsumerLimit	Specify the limit for the number of consumers that are allowed to be idle at any given time.	1	MEDIUM
camel.sink.endpoint.idleTaskExecutionLimit	Specifies the limit for idle executions of a receive task, not having received any message within its execution. If this limit is reached, the task will shut down and leave receiving to other executing tasks (in the case of dynamic scheduling; see the maxConcurrentConsumers setting). There is additional doc available from Spring.	1	MEDIUM
camel.sink.endpoint.includeAllJMSXProperties	Whether to include all JMSXxxx properties when mapping from JMS to Camel Message. Setting this to true will include properties such as JMSXAppID, and JMSXUserID etc. Note: If you are using a custom headerFilterStrategy then this option does not apply.	false	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.jmsKeyFormatStrategy	Pluggable strategy for encoding and decoding JMS keys so they can be compliant with the JMS specification. Camel provides two implementations out of the box: default and passthrough. The default strategy will safely marshal dots and hyphens (. and -). The passthrough strategy leaves the key as is. Can be used for JMS brokers which do not care whether JMS header keys contain illegal characters. You can provide your own implementation of the org.apache.camel.component.jms.JmsKeyFormatStrategy and refer to it using the # notation. One of: [default] [passthrough]	null	MEDIUM
camel.sink.endpoint.mapJmsMessage	Specifies whether Camel should auto map the received JMS message to a suited payload type, such as javax.jms.TextMessage to a String etc.	true	MEDIUM
camel.sink.endpoint.maxMessagesPerTask	The number of messages per task. -1 is unlimited. If you use a range for concurrent consumers (eg min max), then this option can be used to set a value to eg 100 to control how fast the consumers will shrink when less work is required.	-1	MEDIUM
camel.sink.endpoint.messageConverter	To use a custom Spring org.springframework.jms.support.converter.MessageConverter so you can be in control how to map to/from a javax.jms.Message.	null	MEDIUM
camel.sink.endpoint.messageCreatedStrategy	To use the given MessageCreatedStrategy which are invoked when Camel creates new instances of javax.jms.Message objects when Camel is sending a JMS message.	null	MEDIUM
camel.sink.endpoint.messageIdEnabled	When sending, specifies whether message IDs should be added. This is just an hint to the JMS broker. If the JMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.	true	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.messageListenerContainer Factory	Registry ID of the MessageListenerContainerFactory used to determine what org.springframework.jms.listener.AbstractMessageListenerContainer to use to consume messages. Setting this will automatically set consumerType to Custom.	null	MEDIUM
camel.sink.endpoint.messageTimestampEnabled	Specifies whether timestamps should be enabled by default on sending messages. This is just an hint to the JMS broker. If the JMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint the timestamp must be set to its normal value.	true	MEDIUM
camel.sink.endpoint.pubSubNoLocal	Specifies whether to inhibit the delivery of messages published by its own connection.	false	MEDIUM
camel.sink.endpoint.receiveTimeout	The timeout for receiving messages (in milliseconds).	1000L	MEDIUM
camel.sink.endpoint.recoveryInterval	Specifies the interval between recovery attempts, i.e. when a connection is being refreshed, in milliseconds. The default is 5000 ms, that is, 5 seconds.	5000L	MEDIUM
camel.sink.endpoint.requestTimeoutCheckerInterval	Configures how often Camel should check for timed out Exchanges when doing request/reply over JMS. By default Camel checks once per second. But if you must react faster when a timeout occurs, then you can lower this interval, to check more frequently. The timeout is determined by the option requestTimeout.	1000L	MEDIUM
camel.sink.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.transferException	If enabled and you are using Request Reply messaging (InOut) and an Exchange failed on the consumer side, then the caused Exception will be send back in response as a javax.jms.ObjectMessage. If the client is Camel, the returned Exception is rethrown. This allows you to use Camel JMS as a bridge in your routing - for example, using persistent queues to enable robust routing. Notice that if you also have transferExchange enabled, this option takes precedence. The caught exception is required to be serializable. The original Exception on the consumer side can be wrapped in an outer exception such as org.apache.camel.RuntimeCamelException when returned to the producer. Use this with caution as the data is using Java Object serialization and requires the received to be able to deserialize the data at Class level, which forces a strong coupling between the producers and consumer!	false	MEDIUM
camel.sink.endpoint.transferExchange	You can transfer the exchange over the wire instead of just the body and headers. The following fields are transferred: In body, Out body, Fault body, In headers, Out headers, Fault headers, exchange properties, exchange exception. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level. You must enable this option on both the producer and consumer side, so Camel knows the payloads is an Exchange and not a regular payload. Use this with caution as the data is using Java Object serialization and requires the received to be able to deserialize the data at Class level, which forces a strong coupling between the producers and consumer having to use compatible Camel versions!	false	MEDIUM
camel.sink.endpoint.useMessageIDAsCorrelationID	Specifies whether JMSMessageID should always be used as JMSCorrelationID for InOut messages.	false	MEDIUM
camel.sink.endpoint.waitForProvisionCorrelationToBeUpdatedCounter	Number of times to wait for provisional correlation id to be updated to the actual correlation id when doing request/reply over JMS and when the option useMessageIDAsCorrelationID is enabled.	50	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.waitForProvisionCorrelationToBeUpdatedThreadSleepingTime	Interval in millis to sleep each time while waiting for provisional correlation id to be updated.	100L	MEDIUM
camel.sink.endpoint.password	Password to use with the ConnectionFactory. You can also configure username/password directly on the ConnectionFactory.	null	MEDIUM
camel.sink.endpoint.username	Username to use with the ConnectionFactory. You can also configure username/password directly on the ConnectionFactory.	null	MEDIUM
camel.sink.endpoint.transacted	Specifies whether to use transacted mode	false	MEDIUM
camel.sink.endpoint.lazyCreateTransactionManager	If true, Camel will create a JmsTransactionManager, if there is no transactionManager injected when option transacted=true.	true	MEDIUM
camel.sink.endpoint.transactionManager	The Spring transaction manager to use.	null	MEDIUM
camel.sink.endpoint.transactionName	The name of the transaction to use.	null	MEDIUM
camel.sink.endpoint.transactionTimeout	The timeout value of the transaction (in seconds), if using transacted mode.	-1	MEDIUM
camel.component.jms.clientId	Sets the JMS client ID to use. Note that this value, if specified, must be unique and can only be used by a single JMS connection instance. It is typically only required for durable topic subscriptions. If using Apache ActiveMQ you may prefer to use Virtual Topics instead.	null	MEDIUM
camel.component.jms.connectionFactory	The connection factory to be use. A connection factory must be configured either on the component or endpoint.	null	MEDIUM

Name	Description	Default	Priority
camel.component.jms.disableReplyTo	Specifies whether Camel ignores the JMSReplyTo header in messages. If true, Camel does not send a reply back to the destination specified in the JMSReplyTo header. You can use this option if you want Camel to consume from a route and you do not want Camel to automatically send back a reply message because another component in your code handles the reply message. You can also use this option if you want to use Camel as a proxy between different message brokers and you want to route message from one system to another.	false	MEDIUM
camel.component.jms.durableSubscriptionName	The durable subscriber name for specifying durable topic subscriptions. The clientId option must be configured as well.	null	MEDIUM
camel.component.jms.jmsMessageType	Allows you to force the use of a specific javax.jms.Message implementation for sending JMS messages. Possible values are: Bytes, Map, Object, Stream, Text. By default, Camel would determine which JMS message type to use from the In body type. This option allows you to specify it. One of: [Bytes] [Map] [Object] [Stream] [Text]	null	MEDIUM
camel.component.jms.testConnectionOnStartup	Specifies whether to test the connection on startup. This ensures that when Camel starts that all the JMS consumers have a valid connection to the JMS broker. If a connection cannot be granted then Camel throws an exception on startup. This ensures that Camel is not started with failed connections. The JMS producers is tested as well.	false	MEDIUM
camel.component.jms.deliveryDelay	Sets delivery delay to use for send calls for JMS. This option requires JMS 2.0 compliant broker.	-1L	MEDIUM
camel.component.jms.deliveryMode	Specifies the delivery mode to be used. Possibles values are those defined by javax.jms.DeliveryMode. NON_PERSISTENT = 1 and PERSISTENT = 2. One of: [1] [2]	null	MEDIUM
camel.component.jms.deliveryPersistent	Specifies whether persistent delivery is used by default.	true	MEDIUM

Name	Description	Default	Priority
camel.component.jms.explicitQosEnabled	Set if the deliveryMode, priority or timeToLive qualities of service should be used when sending messages. This option is based on Spring's JmsTemplate. The deliveryMode, priority and timeToLive options are applied to the current endpoint. This contrasts with the preserveMessageQos option, which operates at message granularity, reading QoS properties exclusively from the Camel In message headers.	"false"	MEDIUM
camel.component.jms.formatDateHeadersToIso8601	Sets whether JMS date properties should be formatted according to the ISO 8601 standard.	false	MEDIUM
camel.component.jms.lazyStartProducer	Whether the producer should be started lazy (on the first message). By starting lazy you can use this to allow CamelContext and routes to startup in situations where a producer may otherwise fail during starting and cause the route to fail being started. By deferring this startup to be lazy then the startup failure can be handled during routing messages via Camel's routing error handlers. Beware that when the first message is processed then creating and starting the producer may take a little time and prolong the total processing time of the processing.	false	MEDIUM
camel.component.jms.preserveMessageQos	Set to true, if you want to send message using the QoS settings specified on the message, instead of the QoS settings on the JMS endpoint. The following three headers are considered JMSPriority, JMSDeliveryMode, and JMSExpiration. You can provide all or only some of them. If not provided, Camel will fall back to use the values from the endpoint instead. So, when using this option, the headers override the values from the endpoint. The explicitQosEnabled option, by contrast, will only use options set on the endpoint, and not values from the message header.	false	MEDIUM
camel.component.jms.priority	Values greater than 1 specify the message priority when sending (where 0 is the lowest priority and 9 is the highest). The explicitQosEnabled option must also be enabled in order for this option to have any effect. One of: [1] [2] [3] [4] [5] [6] [7] [8] [9]	4	MEDIUM

Name	Description	Default	Priority
camel.component.jms.replyToConcurrentConsumers	Specifies the default number of concurrent consumers when doing request/reply over JMS. See also the maxMessagesPerTask option to control dynamic scaling up/down of threads.	1	MEDIUM
camel.component.jms.replyToMaxConcurrentConsumers	Specifies the maximum number of concurrent consumers when using request/reply over JMS. See also the maxMessagesPerTask option to control dynamic scaling up/down of threads.	null	MEDIUM
camel.component.jms.replyToOnTimeoutMaxConcurrent Consumers	Specifies the maximum number of concurrent consumers for continue routing when timeout occurred when using request/reply over JMS.	1	MEDIUM
camel.component.jms.replyToOverride	Provides an explicit ReplyTo destination in the JMS message, which overrides the setting of replyTo. It is useful if you want to forward the message to a remote Queue and receive the reply message from the ReplyTo destination.	null	MEDIUM
camel.component.jms.replyToType	Allows for explicitly specifying which kind of strategy to use for replyTo queues when doing request/reply over JMS. Possible values are: Temporary, Shared, or Exclusive. By default Camel will use temporary queues. However if replyTo has been configured, then Shared is used by default. This option allows you to use exclusive queues instead of shared ones. See Camel JMS documentation for more details, and especially the notes about the implications if running in a clustered environment, and the fact that Shared reply queues has lower performance than its alternatives Temporary and Exclusive. One of: [Temporary] [Shared] [Exclusive]	null	MEDIUM
camel.component.jms.requestTimeout	The timeout for waiting for a reply when using the InOut Exchange Pattern (in milliseconds). The default is 20 seconds. You can include the header CamelJmsRequestTimeout to override this endpoint configured timeout value, and thus have per message individual timeout values. See also the requestTimeoutCheckerInterval option.	20000L	MEDIUM
camel.component.jms.timeToLive	When sending messages, specifies the time-to-live of the message (in milliseconds).	-1L	MEDIUM

Name	Description	Default	Priority
camel.component.jms.allowAdditionalHeaders	This option is used to allow additional headers which may have values that are invalid according to JMS specification. For example some message systems such as WMQ do this with header names using prefix JMS_IBM_MQMD_ containing values with byte array or other invalid types. You can specify multiple header names separated by comma, and use as suffix for wildcard matching.	null	MEDIUM
camel.component.jms.allowNullBody	Whether to allow sending messages with no body. If this option is false and the message body is null, then an JMSException is thrown.	true	MEDIUM
camel.component.jms.alwaysCopyMessage	If true, Camel will always make a JMS message copy of the message when it is passed to the producer for sending. Copying the message is needed in some situations, such as when a replyToDestinationSelectorName is set (incidentally, Camel will set the alwaysCopyMessage option to true, if a replyToDestinationSelectorName is set)	false	MEDIUM
camel.component.jms.correlationProperty	When using InOut exchange pattern use this JMS property instead of JMSCorrelationID JMS property to correlate messages. If set messages will be correlated solely on the value of this property JMSCorrelationID property will be ignored and not set by Camel.	null	MEDIUM
camel.component.jms.disableTimeToLive	Use this option to force disabling time to live. For example when you do request/reply over JMS, then Camel will by default use the requestTimeout value as time to live on the message being sent. The problem is that the sender and receiver systems have to have their clocks synchronized, so they are in sync. This is not always so easy to archive. So you can use disableTimeToLive=true to not set a time to live value on the sent message. Then the message will not expire on the receiver system. See below in section About time to live for more details.	false	MEDIUM
camel.component.jms.forceSendOriginalMessage	When using mapJmsMessage=false Camel will create a new JMS message to send to a new JMS destination if you touch the headers (get or set) during the route. Set this option to true to force Camel to send the original JMS message that was received.	false	MEDIUM

Name	Description	Default	Priority
camel.component.jms.includeSentJMSMessageID	Only applicable when sending to JMS destination using InOnly (eg fire and forget). Enabling this option will enrich the Camel Exchange with the actual JMSMessageID that was used by the JMS client when the message was sent to the JMS destination.	false	MEDIUM
camel.component.jms.replyToCacheLevelName	Sets the cache level by name for the reply consumer when doing request/reply over JMS. This option only applies when using fixed reply queues (not temporary). Camel will by default use: CACHE_CONSUMER for exclusive or shared w/ replyToSelectorName. And CACHE_SESSION for shared without replyToSelectorName. Some JMS brokers such as IBM WebSphere may require to set the replyToCacheLevelName=CACHE_NONE to work. Note: If using temporary queues then CACHE_NONE is not allowed, and you must use a higher value such as CACHE_CONSUMER or CACHE_SESSION. One of: [CACHE_AUTO] [CACHE_CONNECTION] [CACHE_CONSUMER] [CACHE_NONE] [CACHE_SESSION]	null	MEDIUM
camel.component.jms.replyToDestinationSelectorName	Sets the JMS Selector using the fixed name to be used so you can filter out your own replies from the others when using a shared queue (that is, if you are not using a temporary reply queue).	null	MEDIUM
camel.component.jms.streamMessageTypeEnabled	Sets whether StreamMessage type is enabled or not. Message payloads of streaming kind such as files, InputStream, etc will either by sent as BytesMessage or StreamMessage. This option controls which kind will be used. By default BytesMessage is used which enforces the entire message payload to be read into memory. By enabling this option the message payload is read into memory in chunks and each chunk is then written to the StreamMessage until no more data.	false	MEDIUM
camel.component.jms.allowAutoWiredConnection Factory	Whether to auto-discover ConnectionFactory from the registry, if no connection factory has been configured. If only one instance of ConnectionFactory is found then it will be used. This is enabled by default.	true	MEDIUM

Name	Description	Default	Priority
camel.component.jms.allowAutoWiredDestination Resolver	Whether to auto-discover DestinationResolver from the registry, if no destination resolver has been configured. If only one instance of DestinationResolver is found then it will be used. This is enabled by default.	true	MEDIUM
camel.component.jms.allowSerializedHeaders	Controls whether or not to include serialized headers. Applies only when transferExchange is true. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level.	false	MEDIUM
camel.component.jms.artemisStreamingEnabled	Whether optimizing for Apache Artemis streaming mode.	true	MEDIUM
camel.component.jms.asyncStartListener	Whether to startup the JmsConsumer message listener asynchronously, when starting a route. For example if a JmsConsumer cannot get a connection to a remote JMS broker, then it may block while retrying and/or failover. This will cause Camel to block while starting routes. By setting this option to true, you will let routes startup, while the JmsConsumer connects to the JMS broker using a dedicated thread in asynchronous mode. If this option is used, then beware that if the connection could not be established, then an exception is logged at WARN level, and the consumer will not be able to receive messages; You can then restart the route to retry.	false	MEDIUM
camel.component.jms.asyncStopListener	Whether to stop the JmsConsumer message listener asynchronously, when stopping a route.	false	MEDIUM
camel.component.jms.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.jms.configuration	To use a shared JMS configuration	null	MEDIUM
camel.component.jms.destinationResolver	A pluggable org.springframework.jms.support.destination.DestinationResolver that allows you to use your own resolver (for example, to lookup the real destination in a JNDI registry).	null	MEDIUM

Name	Description	Default	Priority
camel.component.jms.errorHandler	Specifies a org.springframework.util.ErrorHandler to be invoked in case of any uncaught exceptions thrown while processing a Message. By default these exceptions will be logged at the WARN level, if no errorHandler has been configured. You can configure logging level and whether stack traces should be logged using errorHandlerLoggingLevel and errorHandlerLogStackTrace options. This makes it much easier to configure, than having to code a custom errorHandler.	null	MEDIUM
camel.component.jms.exceptionListener	Specifies the JMS Exception Listener that is to be notified of any underlying JMS exceptions.	null	MEDIUM
camel.component.jms.idleConsumerLimit	Specify the limit for the number of consumers that are allowed to be idle at any given time.	1	MEDIUM
camel.component.jms.idleTaskExecutionLimit	Specifies the limit for idle executions of a receive task, not having received any message within its execution. If this limit is reached, the task will shut down and leave receiving to other executing tasks (in the case of dynamic scheduling; see the maxConcurrentConsumers setting). There is additional doc available from Spring.	1	MEDIUM
camel.component.jms.includeAllJMSXProperties	Whether to include all JMSXxxx properties when mapping from JMS to Camel Message. Setting this to true will include properties such as JMSXAppID, and JMSXUserID etc. Note: If you are using a custom headerFilterStrategy then this option does not apply.	false	MEDIUM
camel.component.jms.jmsKeyFormatStrategy	Pluggable strategy for encoding and decoding JMS keys so they can be compliant with the JMS specification. Camel provides two implementations out of the box: default and passthrough. The default strategy will safely marshal dots and hyphens (. and -). The passthrough strategy leaves the key as is. Can be used for JMS brokers which do not care whether JMS header keys contain illegal characters. You can provide your own implementation of the org.apache.camel.component.jms.JmsKeyFormatStrategy and refer to it using the # notation. One of: [default] [passthrough]	null	MEDIUM

Name	Description	Default	Priority
camel.component.jms.mapJmsMessage	Specifies whether Camel should auto map the received JMS message to a suited payload type, such as javax.jms.TextMessage to a String etc.	true	MEDIUM
camel.component.jms.maxMessagesPerTask	The number of messages per task. -1 is unlimited. If you use a range for concurrent consumers (eg min max), then this option can be used to set a value to eg 100 to control how fast the consumers will shrink when less work is required.	-1	MEDIUM
camel.component.jms.messageConverter	To use a custom Spring org.springframework.jms.support.converter.MessageConverter so you can be in control how to map to/from a javax.jms.Message.	null	MEDIUM
camel.component.jms.messageCreatedStrategy	To use the given MessageCreatedStrategy which are invoked when Camel creates new instances of javax.jms.Message objects when Camel is sending a JMS message.	null	MEDIUM
camel.component.jms.messageIdEnabled	When sending, specifies whether message IDs should be added. This is just an hint to the JMS broker. If the JMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.	true	MEDIUM
camel.component.jms.messageListenerContainer Factory	Registry ID of the MessageListenerContainerFactory used to determine what org.springframework.jms.listener.AbstractMessageListenerContainer to use to consume messages. Setting this will automatically set consumerType to Custom.	null	MEDIUM
camel.component.jms.messageTimestampEnabled	Specifies whether timestamps should be enabled by default on sending messages. This is just an hint to the JMS broker. If the JMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint the timestamp must be set to its normal value.	true	MEDIUM
camel.component.jms.pubSubNoLocal	Specifies whether to inhibit the delivery of messages published by its own connection.	false	MEDIUM

Name	Description	Default	Priority
camel.component.jms.queueBrowseStrategy	To use a custom QueueBrowseStrategy when browsing queues	null	MEDIUM
camel.component.jms.receiveTimeout	The timeout for receiving messages (in milliseconds).	1000L	MEDIUM
camel.component.jms.recoveryInterval	Specifies the interval between recovery attempts, i.e. when a connection is being refreshed, in milliseconds. The default is 5000 ms, that is, 5 seconds.	5000L	MEDIUM
camel.component.jms.requestTimeoutCheckerInterval	Configures how often Camel should check for timed out Exchanges when doing request/reply over JMS. By default Camel checks once per second. But if you must react faster when a timeout occurs, then you can lower this interval, to check more frequently. The timeout is determined by the option requestTimeout.	1000L	MEDIUM
camel.component.jms.transferException	If enabled and you are using Request Reply messaging (InOut) and an Exchange failed on the consumer side, then the caused Exception will be send back in response as a javax.jms.ObjectMessage. If the client is Camel, the returned Exception is rethrown. This allows you to use Camel JMS as a bridge in your routing - for example, using persistent queues to enable robust routing. Notice that if you also have transferExchange enabled, this option takes precedence. The caught exception is required to be serializable. The original Exception on the consumer side can be wrapped in an outer exception such as org.apache.camel.RuntimeCamelException when returned to the producer. Use this with caution as the data is using Java Object serialization and requires the received to be able to deserialize the data at Class level, which forces a strong coupling between the producers and consumer!	false	MEDIUM

Name	Description	Default	Priority
camel.component.jms.transferExchange	You can transfer the exchange over the wire instead of just the body and headers. The following fields are transferred: In body, Out body, Fault body, In headers, Out headers, Fault headers, exchange properties, exchange exception. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level. You must enable this option on both the producer and consumer side, so Camel knows the payloads is an Exchange and not a regular payload. Use this with caution as the data is using Java Object serialization and requires the received to be able to deserialize the data at Class level, which forces a strong coupling between the producers and consumer having to use compatible Camel versions!	false	MEDIUM
camel.component.jms.useMessageIDAsCorrelationID	Specifies whether JMSMessageID should always be used as JMSCorrelationID for InOut messages.	false	MEDIUM
camel.component.jms.waitForProvisionCorrelationToBeUpdatedCounter	Number of times to wait for provisional correlation id to be updated to the actual correlation id when doing request/reply over JMS and when the option useMessageIDAsCorrelationID is enabled.	50	MEDIUM
camel.component.jms.waitForProvisionCorrelationToBeUpdatedThreadSleepingTime	Interval in millis to sleep each time while waiting for provisional correlation id to be updated.	100L	MEDIUM
camel.component.jms.headerFilterStrategy	To use a custom org.apache.camel.spi.HeaderFilterStrategy to filter header to and from Camel message.	null	MEDIUM
camel.component.jms.password	Password to use with the ConnectionFactory. You can also configure username/password directly on the ConnectionFactory.	null	MEDIUM
camel.component.jms.username	Username to use with the ConnectionFactory. You can also configure username/password directly on the ConnectionFactory.	null	MEDIUM
camel.component.jms.transacted	Specifies whether to use transacted mode	false	MEDIUM

Name	Description	Default	Priority
camel.component.jms.lazyCreateTransactionManager	If true, Camel will create a JmsTransactionManager, if there is no transactionManager injected when option transacted=true.	true	MEDIUM
camel.component.jms.transactionManager	The Spring transaction manager to use.	null	MEDIUM
camel.component.jms.transactionName	The name of the transaction to use.	null	MEDIUM
camel.component.jms.transactionTimeout	The timeout value of the transaction (in seconds), if using transacted mode.	-1	MEDIUM

3.8. CAMEL-JMS-KAFKA-CONNECTOR SOURCE CONFIGURATION

When using camel-jms-kafka-connector as source make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-jms-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-jms source connector supports 141 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.destinationType	The kind of destination to use One of: [queue] [topic] [temp-queue] [temp-topic]	"queue"	MEDIUM
camel.source.path.destinationName	Name of the queue or topic to use as destination	null	HIGH
camel.source.endpoint.clientId	Sets the JMS client ID to use. Note that this value, if specified, must be unique and can only be used by a single JMS connection instance. It is typically only required for durable topic subscriptions. If using Apache ActiveMQ you may prefer to use Virtual Topics instead.	null	MEDIUM
camel.source.endpoint.connectionFactory	The connection factory to be use. A connection factory must be configured either on the component or endpoint.	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.disableReplyTo	Specifies whether Camel ignores the JMSReplyTo header in messages. If true, Camel does not send a reply back to the destination specified in the JMSReplyTo header. You can use this option if you want Camel to consume from a route and you do not want Camel to automatically send back a reply message because another component in your code handles the reply message. You can also use this option if you want to use Camel as a proxy between different message brokers and you want to route message from one system to another.	false	MEDIUM
camel.source.endpoint.durableSubscriptionName	The durable subscriber name for specifying durable topic subscriptions. The clientId option must be configured as well.	null	MEDIUM
camel.source.endpoint.jmsMessageType	Allows you to force the use of a specific javax.jms.Message implementation for sending JMS messages. Possible values are: Bytes, Map, Object, Stream, Text. By default, Camel would determine which JMS message type to use from the In body type. This option allows you to specify it. One of: [Bytes] [Map] [Object] [Stream] [Text]	null	MEDIUM
camel.source.endpoint.testConnectionOnStartup	Specifies whether to test the connection on startup. This ensures that when Camel starts that all the JMS consumers have a valid connection to the JMS broker. If a connection cannot be granted then Camel throws an exception on startup. This ensures that Camel is not started with failed connections. The JMS producers is tested as well.	false	MEDIUM
camel.source.endpoint.acknowledgementModeName	The JMS acknowledgement name, which is one of: SESSION_TRANSACTED, CLIENT_ACKNOWLEDGE, AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE One of: [SESSION_TRANSACTED] [CLIENT_ACKNOWLEDGE] [AUTO_ACKNOWLEDGE] [DUPS_OK_ACKNOWLEDGE]	"AUTO_ACKNOWLEDGE"	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.asyncConsumer	Whether the JmsConsumer processes the Exchange asynchronously. If enabled then the JmsConsumer may pickup the next message from the JMS queue, while the previous message is being processed asynchronously (by the Asynchronous Routing Engine). This means that messages may be processed not 100% strictly in order. If disabled (as default) then the Exchange is fully processed before the JmsConsumer will pickup the next message from the JMS queue. Note if transacted has been enabled, then asyncConsumer=true does not run asynchronously, as transaction must be executed synchronously (Camel 3.0 may support async transactions).	false	MEDIUM
camel.source.endpoint.autoStartup	Specifies whether the consumer container should auto-startup.	true	MEDIUM
camel.source.endpoint.cacheLevel	Sets the cache level by ID for the underlying JMS resources. See cacheLevelName option for more details.	null	MEDIUM
camel.source.endpoint.cacheLevelName	Sets the cache level by name for the underlying JMS resources. Possible values are: CACHE_AUTO, CACHE_CONNECTION, CACHE_CONSUMER, CACHE_NONE, and CACHE_SESSION. The default setting is CACHE_AUTO. See the Spring documentation and Transactions Cache Levels for more information. One of: [CACHE_AUTO] [CACHE_CONNECTION] [CACHE_CONSUMER] [CACHE_NONE] [CACHE_SESSION]	"CACHE_AUTO"	MEDIUM
camel.source.endpoint.concurrentConsumers	Specifies the default number of concurrent consumers when consuming from JMS (not for request/reply over JMS). See also the maxMessagesPerTask option to control dynamic scaling up/down of threads. When doing request/reply over JMS then the option replyToConcurrentConsumers is used to control number of concurrent consumers on the reply message listener.	1	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.maxConcurrentConsumers	Specifies the maximum number of concurrent consumers when consuming from JMS (not for request/reply over JMS). See also the maxMessagesPerTask option to control dynamic scaling up/down of threads. When doing request/reply over JMS then the option replyToMaxConcurrentConsumers is used to control number of concurrent consumers on the reply message listener.	null	MEDIUM
camel.source.endpoint.replyTo	Provides an explicit ReplyTo destination, which overrides any incoming value of Message.getJMSReplyTo().	null	MEDIUM
camel.source.endpoint.replyToDeliveryPersistent	Specifies whether to use persistent delivery by default for replies.	true	MEDIUM
camel.source.endpoint.selector	Sets the JMS selector to use	null	MEDIUM
camel.source.endpoint.subscriptionDurable	Set whether to make the subscription durable. The durable subscription name to be used can be specified through the subscriptionName property. Default is false. Set this to true to register a durable subscription, typically in combination with a subscriptionName value (unless your message listener class name is good enough as subscription name). Only makes sense when listening to a topic (pub-sub domain), therefore this method switches the pubSubDomain flag as well.	false	MEDIUM
camel.source.endpoint.subscriptionName	Set the name of a subscription to create. To be applied in case of a topic (pub-sub domain) with a shared or durable subscription. The subscription name needs to be unique within this client's JMS client id. Default is the class name of the specified message listener. Note: Only 1 concurrent consumer (which is the default of this message listener container) is allowed for each subscription, except for a shared subscription (which requires JMS 2.0).	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.subscriptionShared	Set whether to make the subscription shared. The shared subscription name to be used can be specified through the subscriptionName property. Default is false. Set this to true to register a shared subscription, typically in combination with a subscriptionName value (unless your message listener class name is good enough as subscription name). Note that shared subscriptions may also be durable, so this flag can (and often will) be combined with subscriptionDurable as well. Only makes sense when listening to a topic (pub-sub domain), therefore this method switches the pubSubDomain flag as well. Requires a JMS 2.0 compatible message broker.	false	MEDIUM
camel.source.endpoint.acceptMessagesWhileStopping	Specifies whether the consumer accept messages while it is stopping. You may consider enabling this option, if you start and stop JMS routes at runtime, while there are still messages enqueued on the queue. If this option is false, and you stop the JMS route, then messages may be rejected, and the JMS broker would have to attempt redeliveries, which yet again may be rejected, and eventually the message may be moved at a dead letter queue on the JMS broker. To avoid this its recommended to enable this option.	false	MEDIUM
camel.source.endpoint.allowReplyManagerQuickStop	Whether the DefaultMessageListenerContainer used in the reply managers for request-reply messaging allow the DefaultMessageListenerContainer.runningAllowed flag to quick stop in case JmsConfiguration#isAcceptMessagesWhileStopping is enabled, and org.apache.camel.CamelContext is currently being stopped. This quick stop ability is enabled by default in the regular JMS consumers but to enable for reply managers you must enable this flag.	false	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.consumerType	The consumer type to use, which can be one of: Simple, Default, or Custom. The consumer type determines which Spring JMS listener to use. Default will use org.springframework.jms.listener.DefaultMessageListenerContainer, Simple will use org.springframework.jms.listener.SimpleMessageListenerContainer. When Custom is specified, the MessageListenerContainerFactory defined by the messageListenerContainerFactory option will determine what org.springframework.jms.listener.AbstractMessageListenerContainer to use. One of: [Simple] [Default] [Custom]	"Default"	MEDIUM
camel.source.endpoint.defaultTaskExecutorType	Specifies what default TaskExecutor type to use in the DefaultMessageListenerContainer, for both consumer endpoints and the ReplyTo consumer of producer endpoints. Possible values: SimpleAsync (uses Spring's SimpleAsyncTaskExecutor) or ThreadPool (uses Spring's ThreadPoolTaskExecutor with optimal values - cached threadpool-like). If not set, it defaults to the previous behaviour, which uses a cached thread pool for consumer endpoints and SimpleAsync for reply consumers. The use of ThreadPool is recommended to reduce thread trash in elastic configurations with dynamically increasing and decreasing concurrent consumers. One of: [ThreadPool] [SimpleAsync]	null	MEDIUM
camel.source.endpoint.eagerLoadingOfProperties	Enables eager loading of JMS properties and payload as soon as a message is loaded which generally is inefficient as the JMS properties may not be required but sometimes can catch early any issues with the underlying JMS provider and the use of JMS properties. See also the option eagerPoisonBody.	false	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.eagerPoisonBody	If eagerLoadingOfProperties is enabled and the JMS message payload (JMS body or JMS properties) is poison (cannot be read/mapped), then set this text as the message body instead so the message can be processed (the cause of the poison are already stored as exception on the Exchange). This can be turned off by setting eagerPoisonBody=false. See also the option eagerLoadingOfProperties.	"Poison JMS message due to ${exception.message}"	MEDIUM
camel.source.endpoint.exceptionHandler	To let the consumer use a custom ExceptionHandler. Notice if the option bridgeErrorHandler is enabled then this option is not in use. By default the consumer will deal with exceptions, that will be logged at WARN or ERROR level and ignored.	null	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	MEDIUM
camel.source.endpoint.exposeListenerSession	Specifies whether the listener session should be exposed when consuming messages.	false	MEDIUM
camel.source.endpoint.replyToSameDestination Allowed	Whether a JMS consumer is allowed to send a reply message to the same destination that the consumer is using to consume from. This prevents an endless loop by consuming and sending back the same message to itself.	false	MEDIUM
camel.source.endpoint.taskExecutor	Allows you to specify a custom task executor for consuming messages.	null	MEDIUM
camel.source.endpoint.allowSerializedHeaders	Controls whether or not to include serialized headers. Applies only when transferExchange is true. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level.	false	MEDIUM
camel.source.endpoint.artemisStreamingEnabled	Whether optimizing for Apache Artemis streaming mode.	true	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.asyncStartListener	Whether to startup the JmsConsumer message listener asynchronously, when starting a route. For example if a JmsConsumer cannot get a connection to a remote JMS broker, then it may block while retrying and/or failover. This will cause Camel to block while starting routes. By setting this option to true, you will let routes startup, while the JmsConsumer connects to the JMS broker using a dedicated thread in asynchronous mode. If this option is used, then beware that if the connection could not be established, then an exception is logged at WARN level, and the consumer will not be able to receive messages; You can then restart the route to retry.	false	MEDIUM
camel.source.endpoint.asyncStopListener	Whether to stop the JmsConsumer message listener asynchronously, when stopping a route.	false	MEDIUM
camel.source.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.source.endpoint.destinationResolver	A pluggable org.springframework.jms.support.destination.DestinationResolver that allows you to use your own resolver (for example, to lookup the real destination in a JNDI registry).	null	MEDIUM
camel.source.endpoint.errorHandler	Specifies a org.springframework.util.ErrorHandler to be invoked in case of any uncaught exceptions thrown while processing a Message. By default these exceptions will be logged at the WARN level, if no errorHandler has been configured. You can configure logging level and whether stack traces should be logged using errorHandlerLoggingLevel and errorHandlerLogStackTrace options. This makes it much easier to configure, than having to code a custom errorHandler.	null	MEDIUM
camel.source.endpoint.exceptionListener	Specifies the JMS Exception Listener that is to be notified of any underlying JMS exceptions.	null	MEDIUM
camel.source.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to filter header to and from Camel message.	null	MEDIUM
camel.source.endpoint.idleConsumerLimit	Specify the limit for the number of consumers that are allowed to be idle at any given time.	1	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.idleTaskExecutionLimit	Specifies the limit for idle executions of a receive task, not having received any message within its execution. If this limit is reached, the task will shut down and leave receiving to other executing tasks (in the case of dynamic scheduling; see the maxConcurrentConsumers setting). There is additional doc available from Spring.	1	MEDIUM
camel.source.endpoint.includeAllJMSXProperties	Whether to include all JMSXxxx properties when mapping from JMS to Camel Message. Setting this to true will include properties such as JMSXAppID, and JMSXUserID etc. Note: If you are using a custom headerFilterStrategy then this option does not apply.	false	MEDIUM
camel.source.endpoint.jmsKeyFormatStrategy	Pluggable strategy for encoding and decoding JMS keys so they can be compliant with the JMS specification. Camel provides two implementations out of the box: default and passthrough. The default strategy will safely marshal dots and hyphens (. and -). The passthrough strategy leaves the key as is. Can be used for JMS brokers which do not care whether JMS header keys contain illegal characters. You can provide your own implementation of the org.apache.camel.component.jms.JmsKeyFormatStrategy and refer to it using the # notation. One of: [default] [passthrough]	null	MEDIUM
camel.source.endpoint.mapJmsMessage	Specifies whether Camel should auto map the received JMS message to a suited payload type, such as javax.jms.TextMessage to a String etc.	true	MEDIUM
camel.source.endpoint.maxMessagesPerTask	The number of messages per task. -1 is unlimited. If you use a range for concurrent consumers (eg min max), then this option can be used to set a value to eg 100 to control how fast the consumers will shrink when less work is required.	-1	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.messageConverter	To use a custom Spring org.springframework.jms.support.converter.MessageConverter so you can be in control how to map to/from a javax.jms.Message.	null	MEDIUM
camel.source.endpoint.messageCreatedStrategy	To use the given MessageCreatedStrategy which are invoked when Camel creates new instances of javax.jms.Message objects when Camel is sending a JMS message.	null	MEDIUM
camel.source.endpoint.messageIdEnabled	When sending, specifies whether message IDs should be added. This is just an hint to the JMS broker. If the JMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.	true	MEDIUM
camel.source.endpoint.messageListenerContainer Factory	Registry ID of the MessageListenerContainerFactory used to determine what org.springframework.jms.listener.AbstractMessageListenerContainer to use to consume messages. Setting this will automatically set consumerType to Custom.	null	MEDIUM
camel.source.endpoint.messageTimestampEnabled	Specifies whether timestamps should be enabled by default on sending messages. This is just an hint to the JMS broker. If the JMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint the timestamp must be set to its normal value.	true	MEDIUM
camel.source.endpoint.pubSubNoLocal	Specifies whether to inhibit the delivery of messages published by its own connection.	false	MEDIUM
camel.source.endpoint.receiveTimeout	The timeout for receiving messages (in milliseconds).	1000L	MEDIUM
camel.source.endpoint.recoveryInterval	Specifies the interval between recovery attempts, i.e. when a connection is being refreshed, in milliseconds. The default is 5000 ms, that is, 5 seconds.	5000L	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.requestTimeoutCheckerInterval	Configures how often Camel should check for timed out Exchanges when doing request/reply over JMS. By default Camel checks once per second. But if you must react faster when a timeout occurs, then you can lower this interval, to check more frequently. The timeout is determined by the option requestTimeout.	1000L	MEDIUM
camel.source.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM
camel.source.endpoint.transferException	If enabled and you are using Request Reply messaging (InOut) and an Exchange failed on the consumer side, then the caused Exception will be send back in response as a javax.jms.ObjectMessage. If the client is Camel, the returned Exception is rethrown. This allows you to use Camel JMS as a bridge in your routing - for example, using persistent queues to enable robust routing. Notice that if you also have transferExchange enabled, this option takes precedence. The caught exception is required to be serializable. The original Exception on the consumer side can be wrapped in an outer exception such as org.apache.camel.RuntimeCamelException when returned to the producer. Use this with caution as the data is using Java Object serialization and requires the received to be able to deserialize the data at Class level, which forces a strong coupling between the producers and consumer!	false	MEDIUM
camel.source.endpoint.transferExchange	You can transfer the exchange over the wire instead of just the body and headers. The following fields are transferred: In body, Out body, Fault body, In headers, Out headers, Fault headers, exchange properties, exchange exception. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level. You must enable this option on both the producer and consumer side, so Camel knows the payloads is an Exchange and not a regular payload. Use this with caution as the data is using Java Object serialization and requires the received to be able to deserialize the data at Class level, which forces a strong coupling between the producers and consumer having to use compatible Camel versions!	false	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.useMessageIDAsCorrelationID	Specifies whether JMSMessageID should always be used as JMSCorrelationID for InOut messages.	false	MEDIUM
camel.source.endpoint.waitForProvisionCorrelationToBeUpdatedCounter	Number of times to wait for provisional correlation id to be updated to the actual correlation id when doing request/reply over JMS and when the option useMessageIDAsCorrelationID is enabled.	50	MEDIUM
camel.source.endpoint.waitForProvisionCorrelationToBeUpdatedThreadSleepingTime	Interval in millis to sleep each time while waiting for provisional correlation id to be updated.	100L	MEDIUM
camel.source.endpoint.errorHandlerLoggingLevel	Allows to configure the default errorHandler logging level for logging uncaught exceptions. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"WARN"	MEDIUM
camel.source.endpoint.errorHandlerLogStackTrace	Allows to control whether stacktraces should be logged or not, by the default errorHandler.	true	MEDIUM
camel.source.endpoint.password	Password to use with the ConnectionFactory. You can also configure username/password directly on the ConnectionFactory.	null	MEDIUM
camel.source.endpoint.username	Username to use with the ConnectionFactory. You can also configure username/password directly on the ConnectionFactory.	null	MEDIUM
camel.source.endpoint.transacted	Specifies whether to use transacted mode	false	MEDIUM
camel.source.endpoint.lazyCreateTransactionManager	If true, Camel will create a JmsTransactionManager, if there is no transactionManager injected when option transacted=true.	true	MEDIUM
camel.source.endpoint.transactionManager	The Spring transaction manager to use.	null	MEDIUM
camel.source.endpoint.transactionName	The name of the transaction to use.	null	MEDIUM
camel.source.endpoint.transactionTimeout	The timeout value of the transaction (in seconds), if using transacted mode.	-1	MEDIUM

Name	Description	Default	Priority
camel.component.jms.clientId	Sets the JMS client ID to use. Note that this value, if specified, must be unique and can only be used by a single JMS connection instance. It is typically only required for durable topic subscriptions. If using Apache ActiveMQ you may prefer to use Virtual Topics instead.	null	MEDIUM
camel.component.jms.connectionFactory	The connection factory to be use. A connection factory must be configured either on the component or endpoint.	null	MEDIUM
camel.component.jms.disableReplyTo	Specifies whether Camel ignores the JMSReplyTo header in messages. If true, Camel does not send a reply back to the destination specified in the JMSReplyTo header. You can use this option if you want Camel to consume from a route and you do not want Camel to automatically send back a reply message because another component in your code handles the reply message. You can also use this option if you want to use Camel as a proxy between different message brokers and you want to route message from one system to another.	false	MEDIUM
camel.component.jms.durableSubscriptionName	The durable subscriber name for specifying durable topic subscriptions. The clientId option must be configured as well.	null	MEDIUM
camel.component.jms.jmsMessageType	Allows you to force the use of a specific javax.jms.Message implementation for sending JMS messages. Possible values are: Bytes, Map, Object, Stream, Text. By default, Camel would determine which JMS message type to use from the In body type. This option allows you to specify it. One of: [Bytes] [Map] [Object] [Stream] [Text]	null	MEDIUM
camel.component.jms.testConnectionOnStartup	Specifies whether to test the connection on startup. This ensures that when Camel starts that all the JMS consumers have a valid connection to the JMS broker. If a connection cannot be granted then Camel throws an exception on startup. This ensures that Camel is not started with failed connections. The JMS producers is tested as well.	false	MEDIUM

Name	Description	Default	Priority
camel.component.jms.acknowledgementModeName	The JMS acknowledgement name, which is one of: SESSION_TRANSACTED, CLIENT_ACKNOWLEDGE, AUTO_ACKNOWLEDGE, DUPS_OK_ACKNOWLEDGE One of: [SESSION_TRANSACTED] [CLIENT_ACKNOWLEDGE] [AUTO_ACKNOWLEDGE] [DUPS_OK_ACKNOWLEDGE]	"AUTO_ACKNOWLEDGE"	MEDIUM
camel.component.jms.asyncConsumer	Whether the JmsConsumer processes the Exchange asynchronously. If enabled then the JmsConsumer may pickup the next message from the JMS queue, while the previous message is being processed asynchronously (by the Asynchronous Routing Engine). This means that messages may be processed not 100% strictly in order. If disabled (as default) then the Exchange is fully processed before the JmsConsumer will pickup the next message from the JMS queue. Note if transacted has been enabled, then asyncConsumer=true does not run asynchronously, as transaction must be executed synchronously (Camel 3.0 may support async transactions).	false	MEDIUM
camel.component.jms.autoStartup	Specifies whether the consumer container should auto-startup.	true	MEDIUM
camel.component.jms.cacheLevel	Sets the cache level by ID for the underlying JMS resources. See cacheLevelName option for more details.	null	MEDIUM
camel.component.jms.cacheLevelName	Sets the cache level by name for the underlying JMS resources. Possible values are: CACHE_AUTO, CACHE_CONNECTION, CACHE_CONSUMER, CACHE_NONE, and CACHE_SESSION. The default setting is CACHE_AUTO. See the Spring documentation and Transactions Cache Levels for more information. One of: [CACHE_AUTO] [CACHE_CONNECTION] [CACHE_CONSUMER] [CACHE_NONE] [CACHE_SESSION]	"CACHE_AUTO"	MEDIUM

Name	Description	Default	Priority
camel.component.jms.concurrentConsumers	Specifies the default number of concurrent consumers when consuming from JMS (not for request/reply over JMS). See also the maxMessagesPerTask option to control dynamic scaling up/down of threads. When doing request/reply over JMS then the option replyToConcurrentConsumers is used to control number of concurrent consumers on the reply message listener.	1	MEDIUM
camel.component.jms.maxConcurrentConsumers	Specifies the maximum number of concurrent consumers when consuming from JMS (not for request/reply over JMS). See also the maxMessagesPerTask option to control dynamic scaling up/down of threads. When doing request/reply over JMS then the option replyToMaxConcurrentConsumers is used to control number of concurrent consumers on the reply message listener.	null	MEDIUM
camel.component.jms.replyTo	Provides an explicit ReplyTo destination, which overrides any incoming value of Message.getJMSReplyTo().	null	MEDIUM
camel.component.jms.replyToDeliveryPersistent	Specifies whether to use persistent delivery by default for replies.	true	MEDIUM
camel.component.jms.selector	Sets the JMS selector to use	null	MEDIUM
camel.component.jms.subscriptionDurable	Set whether to make the subscription durable. The durable subscription name to be used can be specified through the subscriptionName property. Default is false. Set this to true to register a durable subscription, typically in combination with a subscriptionName value (unless your message listener class name is good enough as subscription name). Only makes sense when listening to a topic (pub-sub domain), therefore this method switches the pubSubDomain flag as well.	false	MEDIUM

Name	Description	Default	Priority
camel.component.jms.subscriptionName	Set the name of a subscription to create. To be applied in case of a topic (pub-sub domain) with a shared or durable subscription. The subscription name needs to be unique within this client's JMS client id. Default is the class name of the specified message listener. Note: Only 1 concurrent consumer (which is the default of this message listener container) is allowed for each subscription, except for a shared subscription (which requires JMS 2.0).	null	MEDIUM
camel.component.jms.subscriptionShared	Set whether to make the subscription shared. The shared subscription name to be used can be specified through the subscriptionName property. Default is false. Set this to true to register a shared subscription, typically in combination with a subscriptionName value (unless your message listener class name is good enough as subscription name). Note that shared subscriptions may also be durable, so this flag can (and often will) be combined with subscriptionDurable as well. Only makes sense when listening to a topic (pub-sub domain), therefore this method switches the pubSubDomain flag as well. Requires a JMS 2.0 compatible message broker.	false	MEDIUM
camel.component.jms.acceptMessagesWhileStopping	Specifies whether the consumer accept messages while it is stopping. You may consider enabling this option, if you start and stop JMS routes at runtime, while there are still messages enqueued on the queue. If this option is false, and you stop the JMS route, then messages may be rejected, and the JMS broker would have to attempt redeliveries, which yet again may be rejected, and eventually the message may be moved at a dead letter queue on the JMS broker. To avoid this its recommended to enable this option.	false	MEDIUM

Name	Description	Default	Priority
camel.component.jms.allowReplyManagerQuickStop	Whether the DefaultMessageListenerContainer used in the reply managers for request-reply messaging allow the DefaultMessageListenerContainer.runningAllowed flag to quick stop in case JmsConfiguration#isAcceptMessagesWhileStopping is enabled, and org.apache.camel.CamelContext is currently being stopped. This quick stop ability is enabled by default in the regular JMS consumers but to enable for reply managers you must enable this flag.	false	MEDIUM
camel.component.jms.consumerType	The consumer type to use, which can be one of: Simple, Default, or Custom. The consumer type determines which Spring JMS listener to use. Default will use org.springframework.jms.listener.DefaultMessageListenerContainer, Simple will use org.springframework.jms.listener.SimpleMessageListenerContainer. When Custom is specified, the MessageListenerContainerFactory defined by the messageListenerContainerFactory option will determine what org.springframework.jms.listener.AbstractMessageListenerContainer to use. One of: [Simple] [Default] [Custom]	"Default"	MEDIUM
camel.component.jms.defaultTaskExecutorType	Specifies what default TaskExecutor type to use in the DefaultMessageListenerContainer, for both consumer endpoints and the ReplyTo consumer of producer endpoints. Possible values: SimpleAsync (uses Spring's SimpleAsyncTaskExecutor) or ThreadPool (uses Spring's ThreadPoolTaskExecutor with optimal values - cached threadpool-like). If not set, it defaults to the previous behaviour, which uses a cached thread pool for consumer endpoints and SimpleAsync for reply consumers. The use of ThreadPool is recommended to reduce thread trash in elastic configurations with dynamically increasing and decreasing concurrent consumers. One of: [ThreadPool] [SimpleAsync]	null	MEDIUM

Name	Description	Default	Priority
camel.component.jms.eagerLoadingOfProperties	Enables eager loading of JMS properties and payload as soon as a message is loaded which generally is inefficient as the JMS properties may not be required but sometimes can catch early any issues with the underlying JMS provider and the use of JMS properties. See also the option eagerPoisonBody.	false	MEDIUM
camel.component.jms.eagerPoisonBody	If eagerLoadingOfProperties is enabled and the JMS message payload (JMS body or JMS properties) is poison (cannot be read/mapped), then set this text as the message body instead so the message can be processed (the cause of the poison are already stored as exception on the Exchange). This can be turned off by setting eagerPoisonBody=false. See also the option eagerLoadingOfProperties.	"Poison JMS message due to ${exception.message}"	MEDIUM
camel.component.jms.exposeListenerSession	Specifies whether the listener session should be exposed when consuming messages.	false	MEDIUM
camel.component.jms.replyToSameDestinationAllowed	Whether a JMS consumer is allowed to send a reply message to the same destination that the consumer is using to consume from. This prevents an endless loop by consuming and sending back the same message to itself.	false	MEDIUM
camel.component.jms.taskExecutor	Allows you to specify a custom task executor for consuming messages.	null	MEDIUM
camel.component.jms.allowAutoWiredConnection Factory	Whether to auto-discover ConnectionFactory from the registry, if no connection factory has been configured. If only one instance of ConnectionFactory is found then it will be used. This is enabled by default.	true	MEDIUM
camel.component.jms.allowAutoWiredDestination Resolver	Whether to auto-discover DestinationResolver from the registry, if no destination resolver has been configured. If only one instance of DestinationResolver is found then it will be used. This is enabled by default.	true	MEDIUM
camel.component.jms.allowSerializedHeaders	Controls whether or not to include serialized headers. Applies only when transferExchange is true. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level.	false	MEDIUM

Name	Description	Default	Priority
camel.component.jms.artemisStreamingEnabled	Whether optimizing for Apache Artemis streaming mode.	true	MEDIUM
camel.component.jms.asyncStartListener	Whether to startup the JmsConsumer message listener asynchronously, when starting a route. For example if a JmsConsumer cannot get a connection to a remote JMS broker, then it may block while retrying and/or failover. This will cause Camel to block while starting routes. By setting this option to true, you will let routes startup, while the JmsConsumer connects to the JMS broker using a dedicated thread in asynchronous mode. If this option is used, then beware that if the connection could not be established, then an exception is logged at WARN level, and the consumer will not be able to receive messages; You can then restart the route to retry.	false	MEDIUM
camel.component.jms.asyncStopListener	Whether to stop the JmsConsumer message listener asynchronously, when stopping a route.	false	MEDIUM
camel.component.jms.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.jms.configuration	To use a shared JMS configuration	null	MEDIUM
camel.component.jms.destinationResolver	A pluggable org.springframework.jms.support.destination.DestinationResolver that allows you to use your own resolver (for example, to lookup the real destination in a JNDI registry).	null	MEDIUM
camel.component.jms.errorHandler	Specifies a org.springframework.util.ErrorHandler to be invoked in case of any uncaught exceptions thrown while processing a Message. By default these exceptions will be logged at the WARN level, if no errorHandler has been configured. You can configure logging level and whether stack traces should be logged using errorHandlerLoggingLevel and errorHandlerLogStackTrace options. This makes it much easier to configure, than having to code a custom errorHandler.	null	MEDIUM

Name	Description	Default	Priority
camel.component.jms.exceptionListener	Specifies the JMS Exception Listener that is to be notified of any underlying JMS exceptions.	null	MEDIUM
camel.component.jms.idleConsumerLimit	Specify the limit for the number of consumers that are allowed to be idle at any given time.	1	MEDIUM
camel.component.jms.idleTaskExecutionLimit	Specifies the limit for idle executions of a receive task, not having received any message within its execution. If this limit is reached, the task will shut down and leave receiving to other executing tasks (in the case of dynamic scheduling; see the maxConcurrentConsumers setting). There is additional doc available from Spring.	1	MEDIUM
camel.component.jms.includeAllJMSXProperties	Whether to include all JMSXxxx properties when mapping from JMS to Camel Message. Setting this to true will include properties such as JMSXAppID, and JMSXUserID etc. Note: If you are using a custom headerFilterStrategy then this option does not apply.	false	MEDIUM
camel.component.jms.jmsKeyFormatStrategy	Pluggable strategy for encoding and decoding JMS keys so they can be compliant with the JMS specification. Camel provides two implementations out of the box: default and passthrough. The default strategy will safely marshal dots and hyphens (. and -). The passthrough strategy leaves the key as is. Can be used for JMS brokers which do not care whether JMS header keys contain illegal characters. You can provide your own implementation of the org.apache.camel.component.jms.JmsKeyFormatStrategy and refer to it using the # notation. One of: [default] [passthrough]	null	MEDIUM
camel.component.jms.mapJmsMessage	Specifies whether Camel should auto map the received JMS message to a suited payload type, such as javax.jms.TextMessage to a String etc.	true	MEDIUM
camel.component.jms.maxMessagesPerTask	The number of messages per task. -1 is unlimited. If you use a range for concurrent consumers (eg min max), then this option can be used to set a value to eg 100 to control how fast the consumers will shrink when less work is required.	-1	MEDIUM

Name	Description	Default	Priority
camel.component.jms.messageConverter	To use a custom Spring org.springframework.jms.support.converter.MessageConverter so you can be in control how to map to/from a javax.jms.Message.	null	MEDIUM
camel.component.jms.messageCreatedStrategy	To use the given MessageCreatedStrategy which are invoked when Camel creates new instances of javax.jms.Message objects when Camel is sending a JMS message.	null	MEDIUM
camel.component.jms.messageIdEnabled	When sending, specifies whether message IDs should be added. This is just an hint to the JMS broker. If the JMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.	true	MEDIUM
camel.component.jms.messageListenerContainer Factory	Registry ID of the MessageListenerContainerFactory used to determine what org.springframework.jms.listener.AbstractMessageListenerContainer to use to consume messages. Setting this will automatically set consumerType to Custom.	null	MEDIUM
camel.component.jms.messageTimestampEnabled	Specifies whether timestamps should be enabled by default on sending messages. This is just an hint to the JMS broker. If the JMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint the timestamp must be set to its normal value.	true	MEDIUM
camel.component.jms.pubSubNoLocal	Specifies whether to inhibit the delivery of messages published by its own connection.	false	MEDIUM
camel.component.jms.queueBrowseStrategy	To use a custom QueueBrowseStrategy when browsing queues	null	MEDIUM
camel.component.jms.receiveTimeout	The timeout for receiving messages (in milliseconds).	1000L	MEDIUM
camel.component.jms.recoveryInterval	Specifies the interval between recovery attempts, i.e. when a connection is being refreshed, in milliseconds. The default is 5000 ms, that is, 5 seconds.	5000L	MEDIUM

Name	Description	Default	Priority
camel.component.jms.requestTimeoutCheckerInterval	Configures how often Camel should check for timed out Exchanges when doing request/reply over JMS. By default Camel checks once per second. But if you must react faster when a timeout occurs, then you can lower this interval, to check more frequently. The timeout is determined by the option requestTimeout.	1000L	MEDIUM
camel.component.jms.transferException	If enabled and you are using Request Reply messaging (InOut) and an Exchange failed on the consumer side, then the caused Exception will be send back in response as a javax.jms.ObjectMessage. If the client is Camel, the returned Exception is rethrown. This allows you to use Camel JMS as a bridge in your routing - for example, using persistent queues to enable robust routing. Notice that if you also have transferExchange enabled, this option takes precedence. The caught exception is required to be serializable. The original Exception on the consumer side can be wrapped in an outer exception such as org.apache.camel.RuntimeCamelException when returned to the producer. Use this with caution as the data is using Java Object serialization and requires the received to be able to deserialize the data at Class level, which forces a strong coupling between the producers and consumer!	false	MEDIUM
camel.component.jms.transferExchange	You can transfer the exchange over the wire instead of just the body and headers. The following fields are transferred: In body, Out body, Fault body, In headers, Out headers, Fault headers, exchange properties, exchange exception. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level. You must enable this option on both the producer and consumer side, so Camel knows the payloads is an Exchange and not a regular payload. Use this with caution as the data is using Java Object serialization and requires the received to be able to deserialize the data at Class level, which forces a strong coupling between the producers and consumer having to use compatible Camel versions!	false	MEDIUM

Name	Description	Default	Priority
camel.component.jms.useMessageIDAsCorrelationID	Specifies whether JMSMessageID should always be used as JMSCorrelationID for InOut messages.	false	MEDIUM
camel.component.jms.waitForProvisionCorrelationToBeUpdatedCounter	Number of times to wait for provisional correlation id to be updated to the actual correlation id when doing request/reply over JMS and when the option useMessageIDAsCorrelationID is enabled.	50	MEDIUM
camel.component.jms.waitForProvisionCorrelationToBeUpdatedThreadSleepingTime	Interval in millis to sleep each time while waiting for provisional correlation id to be updated.	100L	MEDIUM
camel.component.jms.headerFilterStrategy	To use a custom org.apache.camel.spi.HeaderFilterStrategy to filter header to and from Camel message.	null	MEDIUM
camel.component.jms.errorHandlerLoggingLevel	Allows to configure the default errorHandler logging level for logging uncaught exceptions. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"WARN"	MEDIUM
camel.component.jms.errorHandlerLogStackTrace	Allows to control whether stacktraces should be logged or not, by the default errorHandler.	true	MEDIUM
camel.component.jms.password	Password to use with the ConnectionFactory. You can also configure username/password directly on the ConnectionFactory.	null	MEDIUM
camel.component.jms.username	Username to use with the ConnectionFactory. You can also configure username/password directly on the ConnectionFactory.	null	MEDIUM
camel.component.jms.transacted	Specifies whether to use transacted mode	false	MEDIUM
camel.component.jms.lazyCreateTransactionManager	If true, Camel will create a JmsTransactionManager, if there is no transactionManager injected when option transacted=true.	true	MEDIUM
camel.component.jms.transactionManager	The Spring transaction manager to use.	null	MEDIUM

Name	Description	Default	Priority
camel.component.jms.transactionName	The name of the transaction to use.	null	MEDIUM
camel.component.jms.transactionTimeout	The timeout value of the transaction (in seconds), if using transacted mode.	-1	MEDIUM

3.9. CAMEL-SALESFORCE-KAFKA-CONNECTOR SOURCE CONFIGURATION

When using camel-salesforce-kafka-connector as source make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-salesforce-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-salesforce source connector supports 116 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.topicName	The name of the topic/channel to use	null	MEDIUM
camel.source.endpoint.apexMethod	APEX method name	null	MEDIUM
camel.source.endpoint.apexQueryParams	Query params for APEX method	null	MEDIUM
camel.source.endpoint.apexUrl	APEX method URL	null	MEDIUM
camel.source.endpoint.apiVersion	Salesforce API version.	"34.0"	MEDIUM
camel.source.endpoint.backoffIncrement	Backoff interval increment for Streaming connection restart attempts for failures beyond CometD auto-reconnect.	1000L	MEDIUM
camel.source.endpoint.batchId	Bulk API Batch ID	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.contentType	Bulk API content type, one of XML, CSV, ZIP_XML, ZIP_CSV One of: [XML] [CSV] [JSON] [ZIP_XML] [ZIP_CSV] [ZIP_JSON]	null	MEDIUM
camel.source.endpoint.defaultReplayId	Default replayId setting if no value is found in initialReplayIdMap	null	MEDIUM
camel.source.endpoint.format	Payload format to use for Salesforce API calls, either JSON or XML, defaults to JSON One of: [JSON] [XML]	null	MEDIUM
camel.source.endpoint.httpClient	Custom Jetty Http Client to use to connect to Salesforce.	null	MEDIUM
camel.source.endpoint.includeDetails	Include details in Salesforce1 Analytics report, defaults to false.	null	MEDIUM
camel.source.endpoint.initialReplayIdMap	Replay IDs to start from per channel name.	null	MEDIUM
camel.source.endpoint.instanceId	Salesforce1 Analytics report execution instance ID	null	MEDIUM
camel.source.endpoint.jobId	Bulk API Job ID	null	MEDIUM
camel.source.endpoint.limit	Limit on number of returned records. Applicable to some of the API, check the Salesforce documentation.	null	MEDIUM
camel.source.endpoint.maxBackoff	Maximum backoff interval for Streaming connection restart attempts for failures beyond CometD auto-reconnect.	30000L	MEDIUM
camel.source.endpoint.notFoundBehaviour	Sets the behaviour of 404 not found status received from Salesforce API. Should the body be set to NULL NotFoundBehaviour#NULL or should a exception be signaled on the exchange NotFoundBehaviour#EXCEPTION - the default. One of: [EXCEPTION] [NULL]	"EXCEPTION"	MEDIUM
camel.source.endpoint.notifyForFields	Notify for fields, options are ALL, REFERENCED, SELECT, WHERE One of: [ALL] [REFERENCED] [SELECT] [WHERE]	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.notifyForOperationCreate	Notify for create operation, defaults to false (API version = 29.0)	null	MEDIUM
camel.source.endpoint.notifyForOperationDelete	Notify for delete operation, defaults to false (API version = 29.0)	null	MEDIUM
camel.source.endpoint.notifyForOperations	Notify for operations, options are ALL, CREATE, EXTENDED, UPDATE (API version 29.0) One of: [ALL] [CREATE] [EXTENDED] [UPDATE]	null	MEDIUM
camel.source.endpoint.notifyForOperationUndelete	Notify for un-delete operation, defaults to false (API version = 29.0)	null	MEDIUM
camel.source.endpoint.notifyForOperationUpdate	Notify for update operation, defaults to false (API version = 29.0)	null	MEDIUM
camel.source.endpoint.objectMapper	Custom Jackson ObjectMapper to use when serializing/deserializing Salesforce objects.	null	MEDIUM
camel.source.endpoint.rawPayload	Use raw payload String for request and response (either JSON or XML depending on format), instead of DTOs, false by default	false	MEDIUM
camel.source.endpoint.reportId	Salesforce1 Analytics report Id	null	MEDIUM
camel.source.endpoint.reportMetadata	Salesforce1 Analytics report metadata for filtering	null	MEDIUM
camel.source.endpoint.resultId	Bulk API Result ID	null	MEDIUM
camel.source.endpoint.sObjectBlobFieldName	SObject blob field name	null	MEDIUM
camel.source.endpoint.sObjectClass	Fully qualified SObject class name, usually generated using camel-salesforce-maven-plugin	null	MEDIUM
camel.source.endpoint.sObjectFields	SObject fields to retrieve	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.sObjectId	SObject ID if required by API	null	MEDIUM
camel.source.endpoint.sObjectIdName	SObject external ID field name	null	MEDIUM
camel.source.endpoint.sObjectIdValue	SObject external ID field value	null	MEDIUM
camel.source.endpoint.sObjectName	SObject name if required or supported by API	null	MEDIUM
camel.source.endpoint.sObjectQuery	Salesforce SOQL query string	null	MEDIUM
camel.source.endpoint.sObjectSearch	Salesforce SOSL search string	null	MEDIUM
camel.source.endpoint.updateTopic	Whether to update an existing Push Topic when using the Streaming API, defaults to false	false	MEDIUM
camel.source.endpoint.bridgeErrorHandler	Allows for bridging the consumer to the Camel routing Error Handler, which mean any exceptions occurred while the consumer is trying to pickup incoming messages, or the likes, will now be processed as a message and handled by the routing Error Handler. By default the consumer will use the org.apache.camel.spi.ExceptionHandler to deal with exceptions, that will be logged at WARN or ERROR level and ignored.	false	MEDIUM
camel.source.endpoint.replayId	The replayId value to use when subscribing	null	MEDIUM
camel.source.endpoint.exceptionHandler	To let the consumer use a custom ExceptionHandler. Notice if the option bridgeErrorHandler is enabled then this option is not in use. By default the consumer will deal with exceptions, that will be logged at WARN or ERROR level and ignored.	null	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	MEDIUM
camel.source.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM
camel.component.salesforce.apexMethod	APEX method name	null	MEDIUM
camel.component.salesforce.apexQueryParams	Query params for APEX method	null	MEDIUM
camel.component.salesforce.apexUrl	APEX method URL	null	MEDIUM
camel.component.salesforce.apiVersion	Salesforce API version.	"34.0"	MEDIUM
camel.component.salesforce.backoffIncrement	Backoff interval increment for Streaming connection restart attempts for failures beyond CometD auto-reconnect.	1000L	MEDIUM
camel.component.salesforce.batchId	Bulk API Batch ID	null	MEDIUM
camel.component.salesforce.contentType	Bulk API content type, one of XML, CSV, ZIP_XML, ZIP_CSV One of: [XML] [CSV] [JSON] [ZIP_XML] [ZIP_CSV] [ZIP_JSON]	null	MEDIUM
camel.component.salesforce.defaultReplayId	Default replayId setting if no value is found in initialReplayIdMap	null	MEDIUM
camel.component.salesforce.format	Payload format to use for Salesforce API calls, either JSON or XML, defaults to JSON One of: [JSON] [XML]	null	MEDIUM
camel.component.salesforce.httpClient	Custom Jetty Http Client to use to connect to Salesforce.	null	MEDIUM
camel.component.salesforce.httpClientConnection Timeout	Connection timeout used by the HttpClient when connecting to the Salesforce server.	60000L	MEDIUM
camel.component.salesforce.httpClientIdleTimeout	Timeout used by the HttpClient when waiting for response from the Salesforce server.	10000L	MEDIUM

Name	Description	Default	Priority
camel.component.salesforce.httpMaxContentLength	Max content length of an HTTP response.	null	MEDIUM
camel.component.salesforce.includeDetails	Include details in Salesforce1 Analytics report, defaults to false.	null	MEDIUM
camel.component.salesforce.initialReplayIdMap	Replay IDs to start from per channel name.	null	MEDIUM
camel.component.salesforce.instanceId	Salesforce1 Analytics report execution instance ID	null	MEDIUM
camel.component.salesforce.jobId	Bulk API Job ID	null	MEDIUM
camel.component.salesforce.limit	Limit on number of returned records. Applicable to some of the API, check the Salesforce documentation.	null	MEDIUM
camel.component.salesforce.maxBackoff	Maximum backoff interval for Streaming connection restart attempts for failures beyond CometD auto-reconnect.	30000L	MEDIUM
camel.component.salesforce.notFoundBehaviour	Sets the behaviour of 404 not found status received from Salesforce API. Should the body be set to NULL NotFoundBehaviour#NULL or should a exception be signaled on the exchange NotFoundBehaviour#EXCEPTION - the default. One of: [EXCEPTION] [NULL]	"EXCEPTION"	MEDIUM
camel.component.salesforce.notifyForFields	Notify for fields, options are ALL, REFERENCED, SELECT, WHERE One of: [ALL] [REFERENCED] [SELECT] [WHERE]	null	MEDIUM
camel.component.salesforce.notifyForOperationCreate	Notify for create operation, defaults to false (API version = 29.0)	null	MEDIUM
camel.component.salesforce.notifyForOperationDelete	Notify for delete operation, defaults to false (API version = 29.0)	null	MEDIUM
camel.component.salesforce.notifyForOperations	Notify for operations, options are ALL, CREATE, EXTENDED, UPDATE (API version 29.0) One of: [ALL] [CREATE] [EXTENDED] [UPDATE]	null	MEDIUM

Name	Description	Default	Priority
camel.component.salesforce.notifyForOperationUndelete	Notify for un-delete operation, defaults to false (API version = 29.0)	null	MEDIUM
camel.component.salesforce.notifyForOperationUpdate	Notify for update operation, defaults to false (API version = 29.0)	null	MEDIUM
camel.component.salesforce.objectMapper	Custom Jackson ObjectMapper to use when serializing/deserializing Salesforce objects.	null	MEDIUM
camel.component.salesforce.packages	In what packages are the generated DTO classes. Typically the classes would be generated using camel-salesforce-maven-plugin. Set it if using the generated DTOs to gain the benefit of using short SObject names in parameters/header values.	null	MEDIUM
camel.component.salesforce.rawPayload	Use raw payload String for request and response (either JSON or XML depending on format), instead of DTOs, false by default	false	MEDIUM
camel.component.salesforce.reportId	Salesforce1 Analytics report Id	null	MEDIUM
camel.component.salesforce.reportMetadata	Salesforce1 Analytics report metadata for filtering	null	MEDIUM
camel.component.salesforce.resultId	Bulk API Result ID	null	MEDIUM
camel.component.salesforce.sObjectBlobFieldName	SObject blob field name	null	MEDIUM
camel.component.salesforce.sObjectClass	Fully qualified SObject class name, usually generated using camel-salesforce-maven-plugin	null	MEDIUM
camel.component.salesforce.sObjectFields	SObject fields to retrieve	null	MEDIUM
camel.component.salesforce.sObjectId	SObject ID if required by API	null	MEDIUM
camel.component.salesforce.sObjectIdName	SObject external ID field name	null	MEDIUM

Name	Description	Default	Priority
camel.component.salesforce.sObjectIdValue	SObject external ID field value	null	MEDIUM
camel.component.salesforce.sObjectName	SObject name if required or supported by API	null	MEDIUM
camel.component.salesforce.sObjectQuery	Salesforce SOQL query string	null	MEDIUM
camel.component.salesforce.sObjectSearch	Salesforce SOSL search string	null	MEDIUM
camel.component.salesforce.updateTopic	Whether to update an existing Push Topic when using the Streaming API, defaults to false	false	MEDIUM
camel.component.salesforce.config	Global endpoint configuration - use to set values that are common to all endpoints	null	MEDIUM
camel.component.salesforce.httpClientProperties	Used to set any properties that can be configured on the underlying HTTP client. Have a look at properties of SalesforceHttpClient and the Jetty HttpClient for all available options.	null	MEDIUM
camel.component.salesforce.longPollingTransport Properties	Used to set any properties that can be configured on the LongPollingTransport used by the BayeuxClient (CometD) used by the streaming api	null	MEDIUM
camel.component.salesforce.bridgeErrorHandler	Allows for bridging the consumer to the Camel routing Error Handler, which mean any exceptions occurred while the consumer is trying to pickup incoming messages, or the likes, will now be processed as a message and handled by the routing Error Handler. By default the consumer will use the org.apache.camel.spi.ExceptionHandler to deal with exceptions, that will be logged at WARN or ERROR level and ignored.	false	MEDIUM
camel.component.salesforce.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.salesforce.httpProxyExcluded Addresses	A list of addresses for which HTTP proxy server should not be used.	null	MEDIUM

Name	Description	Default	Priority
camel.component.salesforce.httpProxyHost	Hostname of the HTTP proxy server to use.	null	MEDIUM
camel.component.salesforce.httpProxyIncluded Addresses	A list of addresses for which HTTP proxy server should be used.	null	MEDIUM
camel.component.salesforce.httpProxyPort	Port number of the HTTP proxy server to use.	null	MEDIUM
camel.component.salesforce.httpProxySocks4	If set to true the configures the HTTP proxy to use as a SOCKS4 proxy.	false	MEDIUM
camel.component.salesforce.authenticationType	Explicit authentication method to be used, one of USERNAME_PASSWORD, REFRESH_TOKEN or JWT. Salesforce component can auto-determine the authentication method to use from the properties set, set this property to eliminate any ambiguity. One of: [USERNAME_PASSWORD] [REFRESH_TOKEN] [JWT]	null	MEDIUM
camel.component.salesforce.clientId	OAuth Consumer Key of the connected app configured in the Salesforce instance setup. Typically a connected app needs to be configured but one can be provided by installing a package.	null	HIGH
camel.component.salesforce.clientSecret	OAuth Consumer Secret of the connected app configured in the Salesforce instance setup.	null	MEDIUM
camel.component.salesforce.httpProxyAuthUri	Used in authentication against the HTTP proxy server, needs to match the URI of the proxy server in order for the httpProxyUsername and httpProxyPassword to be used for authentication.	null	MEDIUM
camel.component.salesforce.httpProxyPassword	Password to use to authenticate against the HTTP proxy server.	null	MEDIUM
camel.component.salesforce.httpProxyRealm	Realm of the proxy server, used in preemptive Basic/Digest authentication methods against the HTTP proxy server.	null	MEDIUM

Name	Description	Default	Priority
camel.component.salesforce.httpProxySecure	If set to false disables the use of TLS when accessing the HTTP proxy.	true	MEDIUM
camel.component.salesforce.httpProxyUseDigestAuth	If set to true Digest authentication will be used when authenticating to the HTTP proxy, otherwise Basic authorization method will be used	false	MEDIUM
camel.component.salesforce.httpProxyUsername	Username to use to authenticate against the HTTP proxy server.	null	MEDIUM
camel.component.salesforce.instanceUrl	URL of the Salesforce instance used after authentication, by default received from Salesforce on successful authentication	null	MEDIUM
camel.component.salesforce.keystore	KeyStore parameters to use in OAuth JWT flow. The KeyStore should contain only one entry with private key and certificate. Salesforce does not verify the certificate chain, so this can easily be a selfsigned certificate. Make sure that you upload the certificate to the corresponding connected app.	null	MEDIUM
camel.component.salesforce.lazyLogin	If set to true prevents the component from authenticating to Salesforce with the start of the component. You would generally set this to the (default) false and authenticate early and be immediately aware of any authentication issues.	false	MEDIUM
camel.component.salesforce.loginConfig	All authentication configuration in one nested bean, all properties set there can be set directly on the component as well	null	MEDIUM
camel.component.salesforce.loginUrl	URL of the Salesforce instance used for authentication, by default set to https://login.salesforce.com	"https://login.salesforce.com"	HIGH
camel.component.salesforce.password	Password used in OAuth flow to gain access to access token. It's easy to get started with password OAuth flow, but in general one should avoid it as it is deemed less secure than other flows. Make sure that you append security token to the end of the password if using one.	null	MEDIUM

Name	Description	Default	Priority
camel.component.salesforce.refreshToken	Refresh token already obtained in the refresh token OAuth flow. One needs to setup a web application and configure a callback URL to receive the refresh token, or configure using the builtin callback at https://login.salesforce.com/services/oauth2/success or https://test.salesforce.com/services/oauth2/success and then retrive the refresh_token from the URL at the end of the flow. Note that in development organizations Salesforce allows hosting the callback web application at localhost.	null	MEDIUM
camel.component.salesforce.sslContextParameters	SSL parameters to use, see SSLContextParameters class for all available options.	null	MEDIUM
camel.component.salesforce.useGlobalSslContext Parameters	Enable usage of global SSL context parameters	false	MEDIUM
camel.component.salesforce.userName	Username used in OAuth flow to gain access to access token. It's easy to get started with password OAuth flow, but in general one should avoid it as it is deemed less secure than other flows.	null	MEDIUM

3.10. CAMEL-SYSLOG-KAFKA-CONNECTOR SOURCE CONFIGURATION

When using camel-syslog-kafka-connector as source make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-syslog-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-syslog source connector supports is based on camel-netty source connector and supports all its options ; however has been already preconfigured and should be sufficient to provide the following properties:

Name	Description	Default	Priority
camel.source.path.protocol	The protocol to use which can be tcp or udp. One of: [tcp] [udp]	null	HIGH
camel.source.path.host	The hostname. For the consumer the hostname is localhost or 0.0.0.0. For the producer the hostname is the remote host to connect to	null	HIGH
camel.source.path.port	The host port number	null	HIGH

3.11. CAMEL-NETTY-KAFKA-CONNECTOR SOURCE CONFIGURATION

When using camel-netty-kafka-connector as source make sure to use the following Maven dependency to have support for the connector:

```
<dependency>
  <groupId>org.apache.camel.kafkaconnector</groupId>
  <artifactId>camel-netty-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

The camel-netty source connector supports 120 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.protocol	The protocol to use which can be tcp or udp. One of: [tcp] [udp]	null	HIGH
camel.source.path.host	The hostname. For the consumer the hostname is localhost or 0.0.0.0. For the producer the hostname is the remote host to connect to	null	HIGH
camel.source.path.port	The host port number	null	HIGH
camel.source.endpoint.disconnect	Whether or not to disconnect(close) from Netty Channel right after use. Can be used for both consumer and producer.	false	MEDIUM
camel.source.endpoint.keepAlive	Setting to ensure socket is not closed due to inactivity	true	MEDIUM
camel.source.endpoint.reuseAddress	Setting to facilitate socket multiplexing	true	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.reuseChannel	This option allows producers and consumers (in client mode) to reuse the same Netty Channel for the lifecycle of processing the Exchange. This is useful if you need to call a server multiple times in a Camel route and want to use the same network connection. When using this, the channel is not returned to the connection pool until the Exchange is done; or disconnected if the disconnect option is set to true. The reused Channel is stored on the Exchange as an exchange property with the key NettyConstants#NETTY_CHANNEL which allows you to obtain the channel during routing and use it as well.	false	MEDIUM
camel.source.endpoint.sync	Setting to set endpoint as one-way or request-response	true	MEDIUM
camel.source.endpoint.tcpNoDelay	Setting to improve TCP protocol performance	true	MEDIUM
camel.source.endpoint.bridgeErrorHandler	Allows for bridging the consumer to the Camel routing Error Handler, which mean any exceptions occurred while the consumer is trying to pickup incoming messages, or the likes, will now be processed as a message and handled by the routing Error Handler. By default the consumer will use the org.apache.camel.spi.ExceptionHandler to deal with exceptions, that will be logged at WARN or ERROR level and ignored.	false	MEDIUM
camel.source.endpoint.broadcast	Setting to choose Multicast over UDP	false	MEDIUM
camel.source.endpoint.clientMode	If the clientMode is true, netty consumer will connect the address as a TCP client.	false	MEDIUM
camel.source.endpoint.reconnect	Used only in clientMode in consumer, the consumer will attempt to reconnect on disconnection if this is enabled	true	MEDIUM
camel.source.endpoint.reconnectInterval	Used if reconnect and clientMode is enabled. The interval in milli seconds to attempt reconnection	10000	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.backlog	Allows to configure a backlog for netty consumer (server). Note the backlog is just a best effort depending on the OS. Setting this option to a value such as 200, 500 or 1000, tells the TCP stack how long the accept queue can be If this option is not configured, then the backlog depends on OS setting.	null	MEDIUM
camel.source.endpoint.bossCount	When netty works on nio mode, it uses default bossCount parameter from Netty, which is 1. User can use this option to override the default bossCount from Netty	1	MEDIUM
camel.source.endpoint.bossGroup	Set the BossGroup which could be used for handling the new connection of the server side across the NettyEndpoint	null	MEDIUM
camel.source.endpoint.disconnectOnNoReply	If sync is enabled then this option dictates NettyConsumer if it should disconnect where there is no reply to send back.	true	MEDIUM
camel.source.endpoint.exceptionHandler	To let the consumer use a custom ExceptionHandler. Notice if the option bridgeErrorHandler is enabled then this option is not in use. By default the consumer will deal with exceptions, that will be logged at WARN or ERROR level and ignored.	null	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	MEDIUM
camel.source.endpoint.nettyServerBootstrapFactory	To use a custom NettyServerBootstrapFactory	null	MEDIUM
camel.source.endpoint.networkInterface	When using UDP then this option can be used to specify a network interface by its name, such as eth0 to join a multicast group.	null	MEDIUM
camel.source.endpoint.noReplyLogLevel	If sync is enabled this option dictates NettyConsumer which logging level to use when logging a there is no reply to send back. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"WARN"	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.serverClosedChannelException CaughtLogLevel	If the server (NettyConsumer) catches an java.nio.channels.ClosedChannelException then its logged using this logging level. This is used to avoid logging the closed channel exceptions, as clients can disconnect abruptly and then cause a flood of closed exceptions in the Netty server. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"DEBUG"	MEDIUM
camel.source.endpoint.serverExceptionCaught Log Level	If the server (NettyConsumer) catches an exception then its logged using this logging level. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"WARN"	MEDIUM
camel.source.endpoint.serverInitializerFactory	To use a custom ServerInitializerFactory	null	MEDIUM
camel.source.endpoint.usingExecutorService	Whether to use ordered thread pool, to ensure events are processed orderly on the same channel.	true	MEDIUM
camel.source.endpoint.allowSerializedHeaders	Only used for TCP when transferExchange is true. When set to true, serializable objects in headers and properties will be added to the exchange. Otherwise Camel will exclude any non-serializable objects and log it at WARN level.	false	MEDIUM
camel.source.endpoint.basicPropertyBinding	Whether the endpoint should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.source.endpoint.channelGroup	To use a explicit ChannelGroup.	null	MEDIUM
camel.source.endpoint.nativeTransport	Whether to use native transport instead of NIO. Native transport takes advantage of the host operating system and is only supported on some platforms. You need to add the netty JAR for the host operating system you are using. See more details at: http://netty.io/wiki/native-transports.html	false	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.options	Allows to configure additional netty options using option. as prefix. For example option.child.keepAlive=false to set the netty option child.keepAlive=false. See the Netty documentation for possible options that can be used.	null	MEDIUM
camel.source.endpoint.receiveBufferSize	The TCP/UDP buffer sizes to be used during inbound communication. Size is bytes.	65536	MEDIUM
camel.source.endpoint.receiveBufferSizePredictor	Configures the buffer size predictor. See details at Jetty documentation and this mail thread.	null	MEDIUM
camel.source.endpoint.sendBufferSize	The TCP/UDP buffer sizes to be used during outbound communication. Size is bytes.	65536	MEDIUM
camel.source.endpoint.synchronous	Sets whether synchronous processing should be strictly used, or Camel is allowed to use asynchronous processing (if supported).	false	MEDIUM
camel.source.endpoint.transferExchange	Only used for TCP. You can transfer the exchange over the wire instead of just the body. The following fields are transferred: In body, Out body, fault body, In headers, Out headers, fault headers, exchange properties, exchange exception. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level.	false	MEDIUM
camel.source.endpoint.udpByteArrayCodec	For UDP only. If enabled the using byte array codec instead of Java serialization protocol.	false	MEDIUM
camel.source.endpoint.workerCount	When netty works on nio mode, it uses default workerCount parameter from Netty (which is cpu_core_threads x 2). User can use this option to override the default workerCount from Netty.	null	MEDIUM
camel.source.endpoint.workerGroup	To use a explicit EventLoopGroup as the boss thread pool. For example to share a thread pool with multiple consumers or producers. By default each consumer or producer has their own worker pool with 2 x cpu count core threads.	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.allowDefaultCodec	The netty component installs a default codec if both, encoder/decoder is null and textline is false. Setting allowDefaultCodec to false prevents the netty component from installing a default codec as the first element in the filter chain.	true	MEDIUM
camel.source.endpoint.autoAppendDelimiter	Whether or not to auto append missing end delimiter when sending using the textline codec.	true	MEDIUM
camel.source.endpoint.decoderMaxLineLength	The max line length to use for the textline codec.	1024	MEDIUM
camel.source.endpoint.decoders	A list of decoders to be used. You can use a String which have values separated by comma, and have the values be looked up in the Registry. Just remember to prefix the value with # so Camel knows it should lookup.	null	MEDIUM
camel.source.endpoint.delimiter	The delimiter to use for the textline codec. Possible values are LINE and NULL. One of: [LINE] [NULL]	"LINE"	MEDIUM
camel.source.endpoint.encoders	A list of encoders to be used. You can use a String which have values separated by comma, and have the values be looked up in the Registry. Just remember to prefix the value with # so Camel knows it should lookup.	null	MEDIUM
camel.source.endpoint.encoding	The encoding (a charset name) to use for the textline codec. If not provided, Camel will use the JVM default Charset.	null	MEDIUM
camel.source.endpoint.textline	Only used for TCP. If no codec is specified, you can use this flag to indicate a text line based codec; if not specified or the value is false, then Object Serialization is assumed over TCP - however only Strings are allowed to be serialized by default.	false	MEDIUM
camel.source.endpoint.enabledProtocols	Which protocols to enable when using SSL	"TLSv1,TLSv1.1,TLSv1.2"	MEDIUM
camel.source.endpoint.keyStoreFile	Client side certificate keystore to be used for encryption	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.keyStoreFormat	Keystore format to be used for payload encryption. Defaults to JKS if not set	null	MEDIUM
camel.source.endpoint.keyStoreResource	Client side certificate keystore to be used for encryption. Is loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	MEDIUM
camel.source.endpoint.needClientAuth	Configures whether the server needs client authentication when using SSL.	false	MEDIUM
camel.source.endpoint.passphrase	Password setting to use in order to encrypt/decrypt payloads sent using SSH	null	MEDIUM
camel.source.endpoint.securityProvider	Security provider to be used for payload encryption. Defaults to SunX509 if not set.	null	MEDIUM
camel.source.endpoint.ssl	Setting to specify whether SSL encryption is applied to this endpoint	false	MEDIUM
camel.source.endpoint.sslClientCertHeaders	When enabled and in SSL mode, then the Netty consumer will enrich the Camel Message with headers having information about the client certificate such as subject name, issuer name, serial number, and the valid date range.	false	MEDIUM
camel.source.endpoint.sslContextParameters	To configure security using SSLContextParameters	null	MEDIUM
camel.source.endpoint.sslHandler	Reference to a class that could be used to return an SSL Handler	null	MEDIUM
camel.source.endpoint.trustStoreFile	Server side certificate keystore to be used for encryption	null	MEDIUM
camel.source.endpoint.trustStoreResource	Server side certificate keystore to be used for encryption. Is loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	MEDIUM
camel.component.netty.configuration	To use the NettyConfiguration as configuration when creating endpoints.	null	MEDIUM
camel.component.netty.disconnect	Whether or not to disconnect(close) from Netty Channel right after use. Can be used for both consumer and producer.	false	MEDIUM

Name	Description	Default	Priority
camel.component.netty .keepAlive	Setting to ensure socket is not closed due to inactivity	true	MEDIUM
camel.component.netty .reuseAddress	Setting to facilitate socket multiplexing	true	MEDIUM
camel.component.netty .reuseChannel	This option allows producers and consumers (in client mode) to reuse the same Netty Channel for the lifecycle of processing the Exchange. This is useful if you need to call a server multiple times in a Camel route and want to use the same network connection. When using this, the channel is not returned to the connection pool until the Exchange is done; or disconnected if the disconnect option is set to true. The reused Channel is stored on the Exchange as an exchange property with the key NettyConstants#NETTY_CHANNEL which allows you to obtain the channel during routing and use it as well.	false	MEDIUM
camel.component.netty .sync	Setting to set endpoint as one-way or request-response	true	MEDIUM
camel.component.netty .tcpNoDelay	Setting to improve TCP protocol performance	true	MEDIUM
camel.component.netty .bridgeErrorHandler	Allows for bridging the consumer to the Camel routing Error Handler, which mean any exceptions occurred while the consumer is trying to pickup incoming messages, or the likes, will now be processed as a message and handled by the routing Error Handler. By default the consumer will use the org.apache.camel.spi.ExceptionHandler to deal with exceptions, that will be logged at WARN or ERROR level and ignored.	false	MEDIUM
camel.component.netty .broadcast	Setting to choose Multicast over UDP	false	MEDIUM
camel.component.netty .clientMode	If the clientMode is true, netty consumer will connect the address as a TCP client.	false	MEDIUM
camel.component.netty .reconnect	Used only in clientMode in consumer, the consumer will attempt to reconnect on disconnection if this is enabled	true	MEDIUM

Name	Description	Default	Priority
camel.component.netty .reconnectInterval	Used if reconnect and clientMode is enabled. The interval in milli seconds to attempt reconnection	10000	MEDIUM
camel.component.netty .backlog	Allows to configure a backlog for netty consumer (server). Note the backlog is just a best effort depending on the OS. Setting this option to a value such as 200, 500 or 1000, tells the TCP stack how long the accept queue can be If this option is not configured, then the backlog depends on OS setting.	null	MEDIUM
camel.component.netty .bossCount	When netty works on nio mode, it uses default bossCount parameter from Netty, which is 1. User can use this option to override the default bossCount from Netty	1	MEDIUM
camel.component.netty .bossGroup	Set the BossGroup which could be used for handling the new connection of the server side across the NettyEndpoint	null	MEDIUM
camel.component.netty .disconnectOnNoReply	If sync is enabled then this option dictates NettyConsumer if it should disconnect where there is no reply to send back.	true	MEDIUM
camel.component.netty .executorService	To use the given EventExecutorGroup.	null	MEDIUM
camel.component.netty .maximumPoolSize	Sets a maximum thread pool size for the netty consumer ordered thread pool. The default size is 2 x cpu_core plus 1. Setting this value to eg 10 will then use 10 threads unless 2 x cpu_core plus 1 is a higher value, which then will override and be used. For example if there are 8 cores, then the consumer thread pool will be 17. This thread pool is used to route messages received from Netty by Camel. We use a separate thread pool to ensure ordering of messages and also in case some messages will block, then nettys worker threads (event loop) wont be affected.	null	MEDIUM
camel.component.netty .nettyServerBootstrap Factory	To use a custom NettyServerBootstrapFactory	null	MEDIUM
camel.component.netty .networkInterface	When using UDP then this option can be used to specify a network interface by its name, such as eth0 to join a multicast group.	null	MEDIUM

Name	Description	Default	Priority
camel.component.netty.noReplyLogLevel	If sync is enabled this option dictates NettyConsumer which logging level to use when logging a there is no reply to send back. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"WARN"	MEDIUM
camel.component.netty.serverClosedChannelExceptionCaughtLogLevel	If the server (NettyConsumer) catches an java.nio.channels.ClosedChannelException then its logged using this logging level. This is used to avoid logging the closed channel exceptions, as clients can disconnect abruptly and then cause a flood of closed exceptions in the Netty server. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"DEBUG"	MEDIUM
camel.component.netty.serverExceptionCaughtLog Level	If the server (NettyConsumer) catches an exception then its logged using this logging level. One of: [TRACE] [DEBUG] [INFO] [WARN] [ERROR] [OFF]	"WARN"	MEDIUM
camel.component.netty.serverInitializerFactory	To use a custom ServerInitializerFactory	null	MEDIUM
camel.component.netty.usingExecutorService	Whether to use ordered thread pool, to ensure events are processed orderly on the same channel.	true	MEDIUM
camel.component.netty.allowSerializedHeaders	Only used for TCP when transferExchange is true. When set to true, serializable objects in headers and properties will be added to the exchange. Otherwise Camel will exclude any non-serializable objects and log it at WARN level.	false	MEDIUM
camel.component.netty.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.netty.channelGroup	To use a explicit ChannelGroup.	null	MEDIUM
camel.component.netty.nativeTransport	Whether to use native transport instead of NIO. Native transport takes advantage of the host operating system and is only supported on some platforms. You need to add the netty JAR for the host operating system you are using. See more details at: http://netty.io/wiki/native-transports.html	false	MEDIUM

Name	Description	Default	Priority
camel.component.netty.options	Allows to configure additional netty options using option. as prefix. For example option.child.keepAlive=false to set the netty option child.keepAlive=false. See the Netty documentation for possible options that can be used.	null	MEDIUM
camel.component.netty.receiveBufferSize	The TCP/UDP buffer sizes to be used during inbound communication. Size is bytes.	65536	MEDIUM
camel.component.netty.receiveBufferSizePredictor	Configures the buffer size predictor. See details at Jetty documentation and this mail thread.	null	MEDIUM
camel.component.netty.sendBufferSize	The TCP/UDP buffer sizes to be used during outbound communication. Size is bytes.	65536	MEDIUM
camel.component.netty.transferExchange	Only used for TCP. You can transfer the exchange over the wire instead of just the body. The following fields are transferred: In body, Out body, fault body, In headers, Out headers, fault headers, exchange properties, exchange exception. This requires that the objects are serializable. Camel will exclude any non-serializable objects and log it at WARN level.	false	MEDIUM
camel.component.netty.udpByteArrayCodec	For UDP only. If enabled the using byte array codec instead of Java serialization protocol.	false	MEDIUM
camel.component.netty.workerCount	When netty works on nio mode, it uses default workerCount parameter from Netty (which is cpu_core_threads x 2). User can use this option to override the default workerCount from Netty.	null	MEDIUM
camel.component.netty.workerGroup	To use a explicit EventLoopGroup as the boss thread pool. For example to share a thread pool with multiple consumers or producers. By default each consumer or producer has their own worker pool with 2 x cpu count core threads.	null	MEDIUM

Name	Description	Default	Priority
camel.component.netty.allowDefaultCodec	The netty component installs a default codec if both, encoder/decoder is null and textline is false. Setting allowDefaultCodec to false prevents the netty component from installing a default codec as the first element in the filter chain.	true	MEDIUM
camel.component.netty.autoAppendDelimiter	Whether or not to auto append missing end delimiter when sending using the textline codec.	true	MEDIUM
camel.component.netty.decoderMaxLineLength	The max line length to use for the textline codec.	1024	MEDIUM
camel.component.netty.decoders	A list of decoders to be used. You can use a String which have values separated by comma, and have the values be looked up in the Registry. Just remember to prefix the value with # so Camel knows it should lookup.	null	MEDIUM
camel.component.netty.delimiter	The delimiter to use for the textline codec. Possible values are LINE and NULL. One of: [LINE] [NULL]	"LINE"	MEDIUM
camel.component.netty.encoders	A list of encoders to be used. You can use a String which have values separated by comma, and have the values be looked up in the Registry. Just remember to prefix the value with # so Camel knows it should lookup.	null	MEDIUM
camel.component.netty.encoding	The encoding (a charset name) to use for the textline codec. If not provided, Camel will use the JVM default Charset.	null	MEDIUM
camel.component.netty.textline	Only used for TCP. If no codec is specified, you can use this flag to indicate a text line based codec; if not specified or the value is false, then Object Serialization is assumed over TCP - however only Strings are allowed to be serialized by default.	false	MEDIUM
camel.component.netty.enabledProtocols	Which protocols to enable when using SSL	"TLSv1,TLSv1.1,TLSv1.2"	MEDIUM
camel.component.netty.keyStoreFile	Client side certificate keystore to be used for encryption	null	MEDIUM

Name	Description	Default	Priority
camel.component.netty .keyStoreFormat	Keystore format to be used for payload encryption. Defaults to JKS if not set	null	MEDIUM
camel.component.netty .keyStoreResource	Client side certificate keystore to be used for encryption. Is loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	MEDIUM
camel.component.netty .needClientAuth	Configures whether the server needs client authentication when using SSL.	false	MEDIUM
camel.component.netty .passphrase	Password setting to use in order to encrypt/decrypt payloads sent using SSH	null	MEDIUM
camel.component.netty .securityProvider	Security provider to be used for payload encryption. Defaults to SunX509 if not set.	null	MEDIUM
camel.component.netty .ssl	Setting to specify whether SSL encryption is applied to this endpoint	false	MEDIUM
camel.component.netty .sslClientCertHeaders	When enabled and in SSL mode, then the Netty consumer will enrich the Camel Message with headers having information about the client certificate such as subject name, issuer name, serial number, and the valid date range.	false	MEDIUM
camel.component.netty .sslContextParameters	To configure security using SSLContextParameters	null	MEDIUM
camel.component.netty .sslHandler	Reference to a class that could be used to return an SSL Handler	null	MEDIUM
camel.component.netty .trustStoreFile	Server side certificate keystore to be used for encryption	null	MEDIUM
camel.component.netty .trustStoreResource	Server side certificate keystore to be used for encryption. Is loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	MEDIUM
camel.component.netty .useGlobalSslContext Parameters	Enable usage of global SSL context parameters.	false	MEDIUM