Red Hat Integration 2020-Q4

Getting Started with Camel Kafka Connector

TECHNOLOGY PREVIEW - Using Camel components as Kafka connectors

Last Updated: 2021-03-11

Red Hat Integration 2020-Q4 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 describes 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 AMQ Streams and Kafka Connect, and is available on both OpenShift Container Platform and Red Hat Enterprise Linux.

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:

1.2.1. Platforms and components

- OpenShift Container Platform 4.5 or 4.6
- Red Hat Enterprise Linux 7.x or 8.x
- AMQ Streams 1.5
- Kafka Connect 2.5
- Camel 3.5
- OpenJDK 8 or 11

1.2.2. Technology Preview features

- Selected Camel Kafka connectors
- Marshaling/unmarshalling of Camel data formats for sink and source connectors
- Aggregation for sink connectors
- Maven archetypes for extending connectors

1.2.3. Camel Kafka connectors

Table 1.1. Camel Kafka connectors in Technology Preview

Connector	Sink/source
Amazon Web Services (AWS2) Kinesis	Sink and source
Amazon Web Services (AWS2) S3	Sink and source
Amazon Web Services (AWS2) SNS	Sink only
Amazon Web Services (AWS2) SQS	Sink and source
Cassandra Query Language (CQL)	Sink and source
Elasticsearch	Sink only
File	Sink only

Connector	Sink/source
Hadoop Distributed File System (HDFS)	Sink only
Hypertext Transfer Protocol (HTTP)	Sink only
Java Database Connectivity (JDBC)	Sink only
Java Message Service (JMS)	Sink and source
MongoDB	Sink and source
Salesforce	Source only
Slack	Source only
Syslog	Source only
Timer	Source only

1.2.4. Camel data formats

The Camel Kafka Connector Technology Preview includes marshaling and unmarshaling of Camel data formats. For example, these formats include Apache Avro, Base64, Google Protobuf, JSON, SOAP, Zip file, and many more. You can configure marshaling and unmarshaling of Camel data formats using properties in your Camel Kafka Connector, configuration.

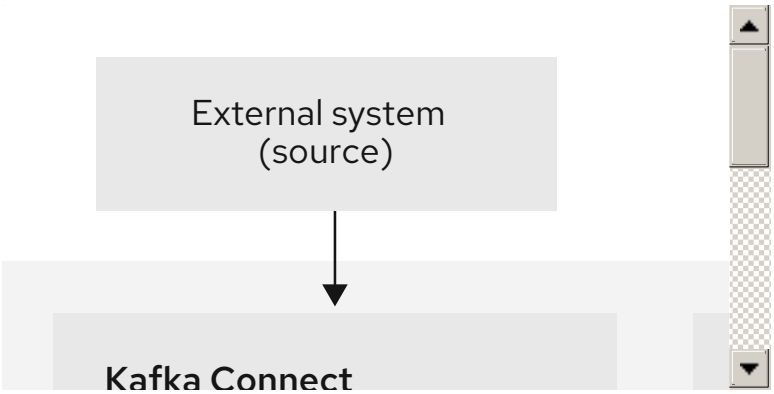
1.3. CAMEL KAFKA CONNECTOR ARCHITECTURE

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:

Figure 1.1. Camel Kafka Connector architecture



Kafka Connect concepts

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, for example, renaming fields or routing to topics based on values.

Aggregator

Sink connectors can use an aggregator to batch up records before sending them to an external system.

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
Source code	Source code for Camel Kafka Connector	Software Downloads > Red Hat Integration
Demonstration examples	Camel Kafka Connector examples and Debezium community example	• Camel Kafka Connector examples • 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 ON OPENSHIFT

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 Source-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 want 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 want 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
```

You should see the following output:

```
secret/PULL_SECRET_NAME created
```

IMPORTANT

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

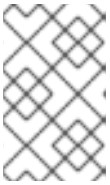
4. Link the secret to your service account to use the secret for pulling images. The following example uses the **default** service account:

```
$ 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 wish 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**.

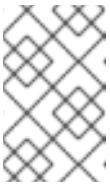
3. In the left navigation menu, click **Operators** > **OperatorHub**.

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** to display the Operator subscription page.

6. Select your subscription settings, for example:

   - **Update Channel** > **stable**
   - **Installation Mode** > **A specific namespace on the cluster**> **myproject**
   - **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 **Install**, and wait a few moments until the Operator is ready for use.

8. 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 .

9. 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 .

10. 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 USING KAFKA CONNECT S2I ON OPENSHIFT

This section explains how to use Kafka Connect Source-2-Image (S2I) with AMQ Streams to add your Camel Kafka connectors to your existing Docker-based Kafka Connect image and to build a new image. This section also shows how to create an instance of a Camel Kafka connector plug-in using an example AWS2 S3 Camel Kafka connector.

Prerequisites

- You must have cluster administrator access to an OpenShift Container Platform cluster.
- 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 Camel Kafka Connector from Software Downloads > Red Hat Integration.
- You must have access to an Amazon S3 bucket.

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 Camel Kafka connectors to the existing Kafka Connect Docker image build, and wait for the new image build to 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
...
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
...
Writing manifest to image destination
Storing signatures
...
Pushing image image-registry.openshift-image-registry.svc:5000/myproject/my-connect-cluster-connect:latest ...
Getting image source signatures
...
Writing manifest to image destination
Storing signatures
Successfully pushed image-registry.openshift-image-registry.svc:5000/myproject/my-connect-cluster-connect@sha256:9db57d33df6d0494ea6ee6e4696fcaf79eb81aabeb0bbc180dec5324d33e7eda
Push successful
```

4. Check that Camel Kafka Connector is available in your Kafka Connect cluster as follows:

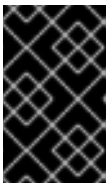
```
$ oc exec -i `oc get pods --field-selector status.phase=Running -l strimzi.io/name=my-connect-cluster-connect -o=jsonpath='{.items[0].metadata.name}'` -- curl -s http://my-connect-cluster-connect-api:8083/connector-plugins
```

You should see something like the following output:

```
[{"class":"org.apache.kafka.connect.file.FileStreamSinkConnector","type":"sink","version":"2.5.0.redhat-00003"},
{"class":"org.apache.kafka.connect.file.FileStreamSourceConnector","type":"source","version":"2.5.0.redhat-00003"},
{"class":"org.apache.kafka.connect.mirror.MirrorCheckpointConnector","type":"source","version":"1"},
{"class":"org.apache.kafka.connect.mirror.MirrorHeartbeatConnector","type":"source","version":"1"},
{"class":"org.apache.kafka.connect.mirror.MirrorSourceConnector","type":"source","version":"1"}]
```

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
kafkaconnects2i.kafka.strimzi.io/my-connect-cluster annotated
```



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 AWS2 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.aws2s3.CamelAws2s3SourceConnector
  tasksMax: 1
  config:
    key.converter: org.apache.kafka.connect.storage.StringConverter
    value.converter: org.apache.kafka.connect.storage.StringConverter
    topics: s3-topic
    camel.source.path.bucketNameOrArn: camel-kafka-connector
    camel.source.maxPollDuration: 10000
    camel.component.aws2-s3.accessKey: xxxx
    camel.component.aws2-s3.secretKey: yyyy
    camel.component.aws2-s3.region: region
EOF
kafkaconnector.kafka.strimzi.io/s3-source-connector created
```

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

```
$ oc exec -i `oc get pods --field-selector status.phase=Running -l strimzi.io/name=my-
connect-cluster-connect -o=jsonpath='{.items[0].metadata.name}'` -- curl -s http://my-
connect-cluster-connect-api:8083/connectors/s3-source-connector/status
```

8. Connect to your AWS Console, and upload a file to the **camel-kafka-connector** AWS S3 bucket to activate the Camel Kafka route.

9. You can run the Kafka console consumer to see the messages received from the topic as follows:

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

Additional resources

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

CHAPTER 3. DEPLOYING CAMEL KAFKA CONNECTOR DEVELOPER EXAMPLES

Camel Kafka Connector provides demonstration examples for selected connectors, which are available from https://github.com/jboss-fuse/camel-kafka-connector-examples. This chapter provides details on how to deploy these examples based on your Camel Kafka Connector installation platform:

- Section 3.1, "Deploying Camel Kafka Connector examples on OpenShift"
- Section 3.2, "Deploying Camel Kafka Connector examples on RHEL"

3.1. DEPLOYING CAMEL KAFKA CONNECTOR EXAMPLES ON OPENSHIFT

This section describes how to deploy Camel Kafka Connector demonstration examples for selected connectors on OpenShift.

Prerequisites

- Scroll down to see the *OpenShift - What is needed* section in each of the readmes shown in the Procedure that follows.

Procedure

1. Go to the GitHub readme for one of the following examples:
   - AWS2 S3 connectors (excluding MinIO source)
   - AWS2 SNS sink connector
   - AWS2 SQS connectors
2. Scroll down to the *OpenShift* section of the readme for your chosen example.
3. Perform the steps described in the readme to run the example.

Additional resources

- Using AMQ Streams on OpenShift

3.2. DEPLOYING CAMEL KAFKA CONNECTOR EXAMPLES ON RHEL

This section describes how to deploy Camel Kafka Connector demonstration examples for selected connectors on Red Hat Enterprise Linux.

Prerequisites

- See the *What is needed* section in each of the readmes shown in the Procedure that follows.

Procedure

1. Go to the GitHub readme for one of the following examples:

- AWS2 S3 connectors
- AWS2 SNS sink connector
- AWS2 SQS connectors
- CQL connectors

2. Perform the steps described in the readme to run the example.

Additional resources

- Using AMQ Streams on RHEL
- Debezium PostgreSQL connector and Apache Camel Kafka Connector (community example)

CHAPTER 4. EXTENDING CAMEL KAFKA CONNECTOR

This chapter explains how to extend and customize Camel Kafka connectors and components. Camel Kafka Connector provides an easy way to configure Camel components directly in the Kafka Connect framework, without needing to write code. However, in some scenarios, you might want to extend and customize Camel Kafka Connector for specific use cases.

4.1. CONFIGURING A CAMEL KAFKA CONNECTOR AGGREGATOR

In some scenarios using a Camel Kafka sink connector, you might want to add an aggregator to batch up your Kafka records before sending them to the external sink system. Typically, this involves defining a specific batch size and timeout for aggregation of records. When complete, the aggregate record is sent to the external system.

You can configure aggregation settings in your Camel Kafka Connector properties using one of the aggregators provided by Apache Camel, or you can implement a custom aggregator in Java. This section describes how to configure the Camel aggregator settings in your Camel Kafka Connector properties.

Prerequisites

- You must have installed Camel Kafka Connector, for example, see Section 2.2, "Installing AMQ Streams and Kafka Connect S2I on OpenShift".
- You must have deployed your sink connector, for example, see Section 2.3, "Deploying Camel Kafka Connector using Kafka Connect S2I on OpenShift". This section shows an example using the AWS S3 sink connector.

Procedure

- Configure your sink connector and aggregator settings in Camel Kafka Connector properties, depending on your installation platform:

  OpenShift

  The following example shows the AWS S3 sink connector and aggregator configuration in a custom resource:

```
oc apply -f - << EOF
apiVersion: kafka.strimzi.io/v1alpha1
kind: KafkaConnector
metadata:
  name: s3-sink-connector
  namespace: myproject
  labels:
    strimzi.io/cluster: my-connect-cluster
spec:
  class: org.apache.camel.kafkaconnector.aws2s3.CamelAws2s3SinkConnector
```

```
  tasksMax: 1
  config:
    key.converter: org.apache.kafka.connect.storage.StringConverter
    value.converter: org.apache.kafka.connect.storage.StringConverter
    topics: s3-topic
    camel.sink.path.bucketNameOrArn: camel-kafka-connector
    camel.sink.endpoint.keyName: ${date:now:yyyyMMdd-HHmmssSSS}-${exchangeId}
    # Camel aggregator settings
    camel.beans.aggregate:
class:org.apache.camel.kafkaconnector.aggregator.StringAggregator
    camel.beans.aggregation.size: 10
    camel.beans.aggregation.timeout: 5000
    camel.component.aws2-s3.accessKey: xxxx
    camel.component.aws2-s3.secretKey: yyyy
    camel.component.aws2-s3.region: region
EOF
```

Red Hat Enterprise Linux

The following example shows the AWS S3 sink connector and aggregator configuration in the **CamelAwss3SinkConnector.properties** file:

```
name=CamelAWS2S3SinkConnector
connector.class=org.apache.camel.kafkaconnector.aws2s3.CamelAws2s3SinkConnector
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter

topics=mytopic

camel.sink.path.bucketNameOrArn=camel-kafka-connector

camel.component.aws2-s3.access-key=xxxx
camel.component.aws2-s3.secret-key=yyyy
camel.component.aws2-s3.region=eu-west-1

camel.sink.endpoint.keyName=${date:now:yyyyMMdd-HHmmssSSS}-${exchangeId}

# Camel aggregator settings
camel.beans.aggregate=class:org.apache.camel.kafkaconnector.aggregator.StringAggregat
or
camel.beans.aggregation.size=10
camel.beans.aggregation.timeout=5000
```

Additional resources

- Demonstration example of AWS2 S3 sink connector with aggregator
- Demonstration example of AWS2 S3 sink connector with zip aggregator
- Apache Camel Kafka Connector aggregation

4.2. WRITING A CUSTOM CAMEL KAFKA CONNECTOR AGGREGATOR

In some scenarios using a Camel Kafka sink connector, you might want to add an aggregator to batch up

your Kafka records before sending them to the external sink system. Typically, this involves defining a specific batch size and timeout for aggregation of records. When complete, the aggregate record is sent to the external system.

You can implement your own aggregator or configure one of the aggregators provided by Apache Camel. This section describes how to implement a custom aggregator in Java using the Camel **AggregationStrategy** class.

Prerequisites

- You must have Red Hat Fuse installed.

Procedure

1. Write your own custom aggregator by implementing the Camel **AggregationStrategy** class, for example:

```
package org.apache.camel.kafkaconnector.aggregator;

import org.apache.camel.AggregationStrategy;
import org.apache.camel.Exchange;
import org.apache.camel.Message;

public class StringAggregator implements AggregationStrategy {

    @Override
    public Exchange aggregate(Exchange oldExchange, Exchange newExchange) { (1)
        // lets append the old body to the new body
        if (oldExchange == null) {
            return newExchange;
        }

        String body = oldExchange.getIn().getBody(String.class); (2)
        if (body != null) {
            Message newIn = newExchange.getIn();
            String newBody = newIn.getBody(String.class);
            if (newBody != null) {
                body += System.lineSeparator() + newBody;
            }

            newIn.setBody(body);
        }
        return newExchange; (3)
    }
}
```

(1) The **oldExchange** and **newExchange** objects correspond to the Kafka records arriving at the aggregator.

(2) In this case, each **newExchange** body will be concatenated with the **oldExchange** body and separated using the **System** line separator.

(3) This process continues until the batch size is completed or the timeout is reached.

2. Add your custom aggregator code to your existing Camel Kafka connector. See Section 4.4, "Extending Camel Kafka connectors using Maven archetypes".

Additional resources

- Apache Camel Kafka Connector aggregators

4.3. CONFIGURING CAMEL DATA FORMATS IN CAMEL KAFKA CONNECTOR

Camel Kafka Connector provides marshaling/unmarshaling of Camel data formats for sink and source connectors. For example, these formats include Apache Avro, Base64, Google Protobuf, JSON, SOAP, Zip file, and many more.

Typically, you would use a Camel **DataFormat** in your Camel DSL to marshal and unmarshal messages to and from different Camel data formats. For example, if you are receiving messages from a Camel File or JMS component and want to unmarshal the payload for further processing, you can use a **DataFormat** to implement this in the Camel DSL.

Using Camel Kafka Connector, you can simply configure marshaling and unmarshaling of Camel data formats using properties in your connector configuration. This section shows how to configure marshaling for the Camel Zip file data format using the **camel.sink.marshal: zipfile** property.

Prerequisites

- You must have Camel Kafka Connector installed on OpenShift or Red Hat Enterprise Linux.
- You must have already built your connector starting from an archetype and edited your **pom.xml** to add the required dependencies. See Section 4.4, "Extending Camel Kafka connectors using Maven archetypes".

Procedure

- Configure the connector settings for marshalling/unmarshalling the data format in your Camel Kafka Connector configuration, depending on your installation platform:

  OpenShift

  The following example shows the AWS S3 sink connector and Camel Zip data format configuration in a custom resource:

```
oc apply -f - << EOF
apiVersion: kafka.strimzi.io/v1alpha1
kind: KafkaConnector
metadata:
  name: s3-sink-connector
  namespace: myproject
  labels:
    strimzi.io/cluster: my-connect-cluster
spec:
  class: org.apache.camel.kafkaconnector.aws2s3.CamelAws2s3SinkConnector
  tasksMax: 1
  config:
    key.converter: org.apache.kafka.connect.storage.StringConverter
    value.converter: org.apache.kafka.connect.storage.StringConverter
```

```
    topics: s3-topic
    camel.sink.path.bucketNameOrArn: camel-kafka-connector
    camel.sink.endpoint.keyName: ${date:now:yyyyMMdd-HHmmssSSS}-
${exchangeId}.zip
    # Camel data format setting
    camel.sink.marshal: zipfile
    camel.component.aws2-s3.accessKey: xxxx
    camel.component.aws2-s3.secretKey: yyyy
    camel.component.aws2-s3.region: region
EOF
```

Red Hat Enterprise Linux

The following example shows the AWS S3 sink connector and Camel Zip data configuration in the **CamelAwss3SinkConnector.properties** file:

```
name=CamelAWS2S3SinkConnector
connector.class=org.apache.camel.kafkaconnector.aws2s3.CamelAws2s3SinkConnector
key.converter=org.apache.kafka.connect.storage.StringConverter
value.converter=org.apache.kafka.connect.storage.StringConverter

topics=mytopic

# Camel data format setting
camel.sink.marshal=zipfile

camel.sink.path.bucketNameOrArn=camel-kafka-connector

camel.component.aws2-s3.access-key=xxxx
camel.component.aws2-s3.secret-key=yyyy
camel.component.aws2-s3.region=eu-west-1

camel.sink.endpoint.keyName=${date:now:yyyyMMdd-HHmmssSSS}-${exchangeId}.zip
```

Additional resources

- Demonstration example of AWS2 S3 sink connector with marshaling of Camel zip data format
- Apache Camel data format types
- Apache Camel DataFormat

4.4. EXTENDING CAMEL KAFKA CONNECTORS USING MAVEN ARCHETYPES

In some scenarios, you might need to extend your Camel Kafka Connector system. For example, when using a sink connector, you might want to add a custom aggregator to batch up your Kafka records before sending them to the external sink system. Alternatively, you might want to configure a connector for marshaling or unmarshaling of Camel data formats, such as Apache Avro, Google Protobuf, JSON, or Zip file.

You can extend an existing Camel Kafka connector using the Maven **camel-kafka-connector-extensible-archetype**. An archetype is a Maven project template, which provides a consistent way of generating a project. This section describes how to use the archetype to create a Maven project to be

extended and how to add your project dependencies.

NOTE

Using Maven archetypes to write additional Kafka Connect converters or transformers is not included in the Technology Preview and has community support only.

Prerequisites

- You must have Apache Maven installed.

Procedure

1. Enter the **mvn archetype:generate** command to create a Maven project to extend Camel Kafka Connector. For example:

```
$ mvn archetype:generate  -
DarchetypeGroupId=org.apache.camel.kafkaconnector.archetypes  -
DarchetypeArtifactId=camel-kafka-connector-extensible-archetype  -
DarchetypeVersion=CONNECTOR_VERSION
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) > generate-sources @
standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) < generate-sources @
standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Interactive mode
[INFO] Archetype repository not defined. Using the one from
[org.apache.camel.kafkaconnector.archetypes:camel-kafka-connector-extensible-
archetype:0.4.0] found in catalog remote
```

2. Enter values for each of the properties when prompted. The following example extends a **camel-aws2-s3-kafka-connector**:

```
Define value for property 'groupId': org.apache.camel.kafkaconnector.extended
Define value for property 'artifactId': myconnector-extended
Define value for property 'version' 1.0-SNAPSHOT: :
Define value for property 'package' org.apache.camel.kafkaconnector.extended: :
Define value for property 'camel-kafka-connector-name': camel-aws2-s3-kafka-connector
[INFO] Using property: camel-kafka-connector-version = CONNECTOR_VERSION
Confirm properties configuration:
groupId: org.apache.camel.kafkaconnector.extended
artifactId: myconnector-extended
version: 1.0-SNAPSHOT
package: org.apache.camel.kafkaconnector.extended
camel-kafka-connector-name: camel-aws2-s3-kafka-connector
camel-kafka-connector-version: CONNECTOR_VERSION
```

3. Enter **Y** to confirm your properties:

```
Y: : Y
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: camel-kafka-connector-extensible-archetype:CONNECTOR_VERSION
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: org.apache.camel.kafkaconnector.extended
[INFO] Parameter: artifactId, Value: myconnector-extended
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: org.apache.camel.kafkaconnector.extended
[INFO] Parameter: packageInPathFormat, Value: org/apache/camel/kafkaconnector/extended
[INFO] Parameter: package, Value: org.apache.camel.kafkaconnector.extended
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: groupId, Value: org.apache.camel.kafkaconnector.extended
[INFO] Parameter: camel-kafka-connector-name, Value: camel-aws2-s3-kafka-connector
[INFO] Parameter: camel-kafka-connector-version, Value: CONNECTOR_VERSION
[INFO] Parameter: artifactId, Value: myconnector-extended
[INFO] Project created from Archetype in dir: /home/workspace/myconnector-extended
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  05:44 min
[INFO] Finished at: 2020-09-04T08:55:00+02:00
[INFO] ------------------------------------------------------------------------
```

4. Enter the dependencies that you need in the **pom.xml** for the created Maven project.

5. Build the Maven project to create a **.zip** or **tar.gz** file for your extended Camel Kafka connector:

```
mvn clean package
```

Additional resources

- Apache Camel Kafka Connector archetypes
- Apache Maven

CHAPTER 5. CAMEL KAFKA CONNECTOR CONFIGURATION 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 5.1. Camel Kafka Connector configuration

Connector	Sink	Source
Amazon Web Services Kinesis	Camel AWS2 Kinesis sink connector	Camel AWS2 Kinesis source connector
Amazon Web Services S3	Camel AWS2 S3 sink connector	Camel AWS2 s3 source connector
Amazon Web Services SNS	Camel AWS2 SNS sink connector	-
Amazon Web Services SQS	Camel AWS2 SQS sink connector	Camel AWS2 SQS source connector
Cassandra Query Language	Camel CQL sink connector	Camel CQL source connector
Elasticsearch	Camel Elasticsearch sink connector	-
File	Camel file sink connector	-
Hadoop Distributed File System	Camel HDFS sink connector	-
Hypertext Transfer Protocol	Camel HTTP sink connector	-
Java Database Connectivity	Camel JDBC sink connector	-
Java Message Service	Camel JMS sink connector	Camel JMS source connector
MongoDB	Camel MongoDB sink connector	Camel MongoDB source connector
Salesforce	-	Camel Salesforce source connector
Slack	-	Camel Slack source connector

Connector	Sink	Source
Syslog	-	Camel syslog source connector
Timer	-	Camel timer source connector

5.1. CAMEL-AWS2-KINESIS-KAFKA-CONNECTOR SINK CONFIGURATION

When using camel-aws2-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-aws2-kinesis-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.aws2kinesis.CamelAws2kinesisSinkConnector
```

The camel-aws2-kinesis sink connector supports 25 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.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking	true	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

Name	Description	Default	Priority
camel.sink.endpoint.region	The region in which Kinesis Firehose client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.sink.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	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.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.aws2-kinesis.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	MEDIUM
camel.component.aws2-kinesis.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking	true	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-kinesis.configuration	Component configuration	null	MEDIUM
camel.component.aws2-kinesis.proxyHost	To define a proxy host when instantiating the Kinesis client	null	MEDIUM
camel.component.aws2-kinesis.proxyPort	To define a proxy port when instantiating the Kinesis client	null	MEDIUM
camel.component.aws2-kinesis.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.component.aws2-kinesis.region	The region in which Kinesis Firehose client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.component.aws2-kinesis.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.component.aws2-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.aws2-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.aws2-kinesis.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.component.aws2-kinesis.secretKey	Amazon AWS Secret Key	null	MEDIUM

The camel-aws2-kinesis sink connector has no converters out of the box.

The camel-aws2-kinesis sink connector has no transforms out of the box.

The camel-aws2-kinesis sink connector has no aggregation strategies out of the box.

5.2. CAMEL-AWS2-KINESIS-KAFKA-CONNECTOR SOURCE CONFIGURATION

When using camel-aws2-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-aws2-kinesis-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.aws2kinesis.CamelAws2kinesisSourceConnector
```

The camel-aws2-kinesis source connector supports 53 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.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking	true	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 Firehose client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	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.iteratorType	Defines where in the Kinesis stream to start getting records One of: [AT_SEQUENCE_NUMBER] [AFTER_SEQUENCE_NUMBER] [TRIM_HORIZON] [LATEST] [AT_TIMESTAMP] [null]	"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

Name	Description	Default	Priority
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.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

Name	Description	Default	Priority
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. Use value spring or quartz for built in scheduler	"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

Name	Description	Default	Priority
camel.source.endpoint.secretKey	Amazon AWS Secret Key	null	MEDIUM
camel.component.aws2-kinesis.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	MEDIUM
camel.component.aws2-kinesis.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking	true	MEDIUM
camel.component.aws2-kinesis.configuration	Component configuration	null	MEDIUM
camel.component.aws2-kinesis.proxyHost	To define a proxy host when instantiating the Kinesis client	null	MEDIUM
camel.component.aws2-kinesis.proxyPort	To define a proxy port when instantiating the Kinesis client	null	MEDIUM
camel.component.aws2-kinesis.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.component.aws2-kinesis.region	The region in which Kinesis Firehose client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.component.aws2-kinesis.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.component.aws2-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

Name	Description	Default	Priority
camel.component.aws2-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] [null]	"TRIM_HORIZON"	MEDIUM
camel.component.aws2-kinesis.maxResultsPerRequest	Maximum number of records that will be fetched in each poll	1	MEDIUM
camel.component.aws2-kinesis.sequenceNumber	The sequence number to start polling from. Required if iteratorType is set to AFTER_SEQUENCE_NUMBER or AT_SEQUENCE_NUMBER	null	MEDIUM
camel.component.aws2-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.aws2-kinesis.shardId	Defines which shardId in the Kinesis stream to get records from	null	MEDIUM
camel.component.aws2-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.aws2-kinesis.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.component.aws2-kinesis.secretKey	Amazon AWS Secret Key	null	MEDIUM

The camel-aws2-kinesis sink connector has no converters out of the box.

The camel-aws2-kinesis sink connector has no transforms out of the box.

The camel-aws2-kinesis sink connector has no aggregation strategies out of the box.

5.3. CAMEL-AWS2-S3-KAFKA-CONNECTOR SINK CONFIGURATION

When using camel-aws2-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-aws2-s3-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.aws2s3.CamelAws2s3SinkConnector
```

The camel-aws2-s3 sink connector supports 61 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 S3 bucket bucketName. This will apply also in case of moveAfterRead option enabled and it will create the destinationBucket if it doesn't exist already.	true	MEDIUM
camel.sink.endpoint.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM
camel.sink.endpoint.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	MEDIUM
camel.sink.endpoint.pojoRequest	If we want to use a POJO request as body or not	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 SQS client	null	MEDIUM

Name	Description	Default	Priority
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 lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.sink.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.sink.endpoint.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	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.customerAlgorithm	Define the customer algorithm to use in case CustomerKey is enabled	null	MEDIUM
camel.sink.endpoint.customerKeyId	Define the id of Customer key to use in case CustomerKey is enabled	null	MEDIUM
camel.sink.endpoint.customerKeyMD5	Define the MD5 of Customer key to use in case CustomerKey is enabled	null	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] [listObjects] [deleteObject] [deleteBucket] [listBuckets] [getObject] [getObjectRange]	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.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.useCustomerKey	Define if Customer Key must be used or not	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

Name	Description	Default	Priority
camel.sink.endpoint.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.sink.endpoint.secretKey	Amazon AWS Secret Key	null	MEDIUM
camel.component.aws2-s3.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	MEDIUM
camel.component.aws2-s3.autoCreateBucket	Setting the autocreation of the S3 bucket bucketName. This will apply also in case of moveAfterRead option enabled and it will create the destinationBucket if it doesn't exist already.	true	MEDIUM
camel.component.aws2-s3.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM
camel.component.aws2-s3.configuration	The component configuration	null	MEDIUM
camel.component.aws2-s3.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	MEDIUM
camel.component.aws2-s3.pojoRequest	If we want to use a POJO request as body or not	false	MEDIUM
camel.component.aws2-s3.policy	The policy for this queue to set in the com.amazonaws.services.s3.AmazonS3#setBucketPolicy() method.	null	MEDIUM
camel.component.aws2-s3.proxyHost	To define a proxy host when instantiating the SQS client	null	MEDIUM
camel.component.aws2-s3.proxyPort	Specify a proxy port to be used inside the client definition.	null	MEDIUM
camel.component.aws2-s3.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-s3.region	The region in which S3 client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.component.aws2-s3.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.component.aws2-s3.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	MEDIUM
camel.component.aws2-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.aws2-s3.customerAlgorithm	Define the customer algorithm to use in case CustomerKey is enabled	null	MEDIUM
camel.component.aws2-s3.customerKeyId	Define the id of Customer key to use in case CustomerKey is enabled	null	MEDIUM
camel.component.aws2-s3.customerKeyMD5	Define the MD5 of Customer key to use in case CustomerKey is enabled	null	MEDIUM
camel.component.aws2-s3.deleteAfterWrite	Delete file object after the S3 file has been uploaded	false	MEDIUM
camel.component.aws2-s3.keyName	Setting the key name for an element in the bucket through endpoint parameter	null	MEDIUM
camel.component.aws2-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.aws2-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.aws2-s3.operation	The operation to do in case the user don't want to do only an upload One of: [copyObject] [listObjects] [deleteObject] [deleteBucket] [listBuckets] [getObject] [getObjectRange]	null	MEDIUM
camel.component.aws2-s3.partSize	Setup the partSize which is used in multi part upload, the default size is 25M.	26214400L	MEDIUM
camel.component.aws2-s3.storageClass	The storage class to set in the com.amazonaws.services.s3.model.PutObjectRequest request.	null	MEDIUM
camel.component.aws2-s3.awsKMSKeyId	Define the id of KMS key to use in case KMS is enabled	null	MEDIUM
camel.component.aws2-s3.useAwsKMS	Define if KMS must be used or not	false	MEDIUM
camel.component.aws2-s3.useCustomerKey	Define if Customer Key must be used or not	false	MEDIUM
camel.component.aws2-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.aws2-s3.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.component.aws2-s3.secretKey	Amazon AWS Secret Key	null	MEDIUM

The camel-aws2-s3 sink connector supports 1 converters out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.aws2s3.converters.S3ObjectConverter
```

The camel-aws2-s3 sink connector has no transforms out of the box.

The camel-aws2-s3 sink connector has no aggregation strategies out of the box.

5.4. CAMEL-AWS2-S3-KAFKA-CONNECTOR SOURCE CONFIGURATION

When using camel-aws2-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-aws2-s3-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.aws2s3.CamelAws2s3SourceConnector
```

The camel-aws2-s3 source connector supports 85 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 S3 bucket bucketName. This will apply also in case of moveAfterRead option enabled and it will create the destinationBucket if it doesn't exist already.	true	MEDIUM
camel.source.endpoint.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM
camel.source.endpoint.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	MEDIUM
camel.source.endpoint.pojoRequest	If we want to use a POJO request as body or not	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 SQS 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 lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.source.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.source.endpoint.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	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.customerAlgorithm	Define the customer algorithm to use in case CustomerKey is enabled	null	MEDIUM
camel.source.endpoint.customerKeyId	Define the id of Customer key to use in case CustomerKey is enabled	null	MEDIUM
camel.source.endpoint.customerKeyMD5	Define the MD5 of Customer key to use in case CustomerKey is enabled	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.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 AWS2S3Constants#BUCKET_NAME and AWS2S3Constants#KEY headers, or only the AWS2S3Constants#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
camel.source.endpoint.destinationBucket	Define the destination bucket where an object must be moved when moveAfterRead is set to true.	null	MEDIUM
camel.source.endpoint.destinationBucketPrefix	Define the destination bucket prefix to use when an object must be moved and moveAfterRead is set to true.	null	MEDIUM
camel.source.endpoint.destinationBucketSuffix	Define the destination bucket suffix to use when an object must be moved and moveAfterRead is set to true.	null	MEDIUM
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.includeFolders	If it is true, the folders/directories will be consumed. If it is false, they will be ignored, and Exchanges will not be created for those	true	MEDIUM

Name	Description	Default	Priority
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.moveAfterRead	Move objects from S3 bucket to a different bucket after they have been retrieved. To accomplish the operation the destinationBucket option must be set. The copy bucket operation is only performed if the Exchange is committed. If a rollback occurs, the object is not moved.	false	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
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. Use value spring or quartz for built in scheduler	"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.aws2-s3.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	MEDIUM
camel.component.aws2-s3.autoCreateBucket	Setting the autocreation of the S3 bucket bucketName. This will apply also in case of moveAfterRead option enabled and it will create the destinationBucket if it doesn't exist already.	true	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-s3.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM
camel.component.aws2-s3.configuration	The component configuration	null	MEDIUM
camel.component.aws2-s3.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	MEDIUM
camel.component.aws2-s3.pojoRequest	If we want to use a POJO request as body or not	false	MEDIUM
camel.component.aws2-s3.policy	The policy for this queue to set in the com.amazonaws.services.s3.AmazonS3#setBucketPolicy() method.	null	MEDIUM
camel.component.aws2-s3.proxyHost	To define a proxy host when instantiating the SQS client	null	MEDIUM
camel.component.aws2-s3.proxyPort	Specify a proxy port to be used inside the client definition.	null	MEDIUM
camel.component.aws2-s3.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.component.aws2-s3.region	The region in which S3 client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.component.aws2-s3.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.component.aws2-s3.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	MEDIUM
camel.component.aws2-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.aws2-s3.customerAlgorithm	Define the customer algorithm to use in case CustomerKey is enabled	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-s3.customerKeyId	Define the id of Customer key to use in case CustomerKey is enabled	null	MEDIUM
camel.component.aws2-s3.customerKeyMD5	Define the MD5 of Customer key to use in case CustomerKey is enabled	null	MEDIUM
camel.component.aws2-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.aws2-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 AWS2S3Constants#BUCKET_NAME and AWS2S3Constants#KEY headers, or only the AWS2S3Constants#KEY header.	true	MEDIUM
camel.component.aws2-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.aws2-s3.destinationBucket	Define the destination bucket where an object must be moved when moveAfterRead is set to true.	null	MEDIUM
camel.component.aws2-s3.destinationBucketPrefix	Define the destination bucket prefix to use when an object must be moved and moveAfterRead is set to true.	null	MEDIUM
camel.component.aws2-s3.destinationBucketSuffix	Define the destination bucket suffix to use when an object must be moved and moveAfterRead is set to true.	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-s3.fileName	To get the object from the bucket with the given file name	null	MEDIUM
camel.component.aws2-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.aws2-s3.includeFolders	If it is true, the folders/directories will be consumed. If it is false, they will be ignored, and Exchanges will not be created for those	true	MEDIUM
camel.component.aws2-s3.moveAfterRead	Move objects from S3 bucket to a different bucket after they have been retrieved. To accomplish the operation the destinationBucket option must be set. The copy bucket operation is only performed if the Exchange is committed. If a rollback occurs, the object is not moved.	false	MEDIUM
camel.component.aws2-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.aws2-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.aws2-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.aws2-s3.accessKey	Amazon AWS Access Key	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-s3.secretKey	Amazon AWS Secret Key	null	MEDIUM

The camel-aws2-s3 sink connector supports 1 converters out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.aws2s3.converters.S3ObjectConverter
```

The camel-aws2-s3 sink connector has no transforms out of the box.

The camel-aws2-s3 sink connector has no aggregation strategies out of the box.

5.5. CAMEL-AWS2-SNS-KAFKA-CONNECTOR SINK CONFIGURATION

When using camel-aws2-sns-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-aws2-sns-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.aws2sns.CamelAws2snsSinkConnector
```

The camel-aws2-sns sink connector supports 42 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.topicNameOrArn	Topic name or ARN	null	HIGH
camel.sink.endpoint.amazonSNSClient	To use the AmazonSNS as the client	null	MEDIUM
camel.sink.endpoint.autoCreateTopic	Setting the autocreation of the topic	true	MEDIUM
camel.sink.endpoint.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM
camel.sink.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to map headers to/from Camel.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.kmsMasterKeyId	The ID of an AWS-managed customer master key (CMK) for Amazon SNS or a custom CMK.	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.messageStructure	The message structure to use such as json	null	MEDIUM
camel.sink.endpoint.policy	The policy for this queue	null	MEDIUM
camel.sink.endpoint.proxyHost	To define a proxy host when instantiating the SNS client	null	MEDIUM
camel.sink.endpoint.proxyPort	To define a proxy port when instantiating the SNS client	null	MEDIUM
camel.sink.endpoint.proxyProtocol	To define a proxy protocol when instantiating the SNS client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.sink.endpoint.queueUrl	The queueUrl to subscribe to	null	MEDIUM
camel.sink.endpoint.region	The region in which SNS client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.sink.endpoint.serverSideEncryptionEnabled	Define if Server Side Encryption is enabled or not on the topic	false	MEDIUM
camel.sink.endpoint.subject	The subject which is used if the message header 'CamelAwsSnsSubject' is not present.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.subscribeSNStoSQS	Define if the subscription between SNS Topic and SQS must be done or not	false	MEDIUM
camel.sink.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	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.aws2-sns.amazonSNSClient	To use the AmazonSNS as the client	null	MEDIUM
camel.component.aws2-sns.autoCreateTopic	Setting the autocreation of the topic	true	MEDIUM
camel.component.aws2-sns.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM
camel.component.aws2-sns.configuration	Component configuration	null	MEDIUM
camel.component.aws2-sns.kmsMasterKeyId	The ID of an AWS-managed customer master key (CMK) for Amazon SNS or a custom CMK.	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-sns.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.aws2-sns.messageStructure	The message structure to use such as json	null	MEDIUM
camel.component.aws2-sns.policy	The policy for this queue	null	MEDIUM
camel.component.aws2-sns.proxyHost	To define a proxy host when instantiating the SNS client	null	MEDIUM
camel.component.aws2-sns.proxyPort	To define a proxy port when instantiating the SNS client	null	MEDIUM
camel.component.aws2-sns.proxyProtocol	To define a proxy protocol when instantiating the SNS client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.component.aws2-sns.queueUrl	The queueUrl to subscribe to	null	MEDIUM
camel.component.aws2-sns.region	The region in which SNS client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.component.aws2-sns.serverSideEncryption Enabled	Define if Server Side Encryption is enabled or not on the topic	false	MEDIUM
camel.component.aws2-sns.subject	The subject which is used if the message header 'CamelAwsSnsSubject' is not present.	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-sns.subscribeSNStoSQS	Define if the subscription between SNS Topic and SQS must be done or not	false	MEDIUM
camel.component.aws2-sns.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.component.aws2-sns.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.aws2-sns.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.component.aws2-sns.secretKey	Amazon AWS Secret Key	null	MEDIUM

The camel-aws2-sns sink connector has no converters out of the box.

The camel-aws2-sns sink connector has no transforms out of the box.

The camel-aws2-sns sink connector has no aggregation strategies out of the box.

5.6. CAMEL-AWS2-SQS-KAFKA-CONNECTOR SINK CONFIGURATION

When using camel-aws2-sqs-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-aws2-sqs-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.aws2sqs.CamelAws2sqsSinkConnector
```

The camel-aws2-sqs sink connector supports 56 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.queueNameOrArn	Queue name or ARN	null	HIGH

Name	Description	Default	Priority
camel.sink.endpoint.amazonAWSHost	The hostname of the Amazon AWS cloud.	"amazonaws.com"	MEDIUM
camel.sink.endpoint.amazonSQSClient	To use the AmazonSQS as client	null	MEDIUM
camel.sink.endpoint.autoCreateQueue	Setting the autocreation of the queue	true	MEDIUM
camel.sink.endpoint.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM
camel.sink.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to map headers to/from Camel.	null	MEDIUM
camel.sink.endpoint.protocol	The underlying protocol used to communicate with SQS	"https"	MEDIUM
camel.sink.endpoint.proxyProtocol	To define a proxy protocol when instantiating the SQS client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.sink.endpoint.queueOwnerAWSAccountId	Specify the queue owner aws account id when you need to connect the queue with different account owner.	null	MEDIUM
camel.sink.endpoint.region	The region in which SQS client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.sink.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.sink.endpoint.delaySeconds	Delay sending messages for a number of seconds.	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.messageDeduplicationIdStrategy	Only for FIFO queues. Strategy for setting the messageDeduplicationId on the message. Can be one of the following options: useExchangeId, useContentBasedDeduplication. For the useContentBasedDeduplication option, no messageDeduplicationId will be set on the message. One of: [useExchangeId] [useContentBasedDeduplication]	"useExchangeId"	MEDIUM
camel.sink.endpoint.messageGroupIdStrategy	Only for FIFO queues. Strategy for setting the messageGroupId on the message. Can be one of the following options: useConstant, useExchangeId, usePropertyValue. For the usePropertyValue option, the value of property CamelAwsMessageGroupId will be used. One of: [useConstant] [useExchangeId] [usePropertyValue]	null	MEDIUM
camel.sink.endpoint.operation	The operation to do in case the user don't want to send only a message One of: [sendBatchMessage] [deleteMessage] [listQueues] [purgeQueue]	null	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.delayQueue	Define if you want to apply delaySeconds option to the queue or on single messages	false	MEDIUM
camel.sink.endpoint.queueUrl	To define the queueUrl explicitly. All other parameters, which would influence the queueUrl, are ignored. This parameter is intended to be used, to connect to a mock implementation of SQS, for testing purposes.	null	MEDIUM

Name	Description	Default	Priority
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.proxyHost	To define a proxy host when instantiating the SQS client	null	MEDIUM
camel.sink.endpoint.proxyPort	To define a proxy port when instantiating the SQS client	null	MEDIUM
camel.sink.endpoint.maximumMessageSize	The maximumMessageSize (in bytes) an SQS message can contain for this queue.	null	MEDIUM
camel.sink.endpoint.messageRetentionPeriod	The messageRetentionPeriod (in seconds) a message will be retained by SQS for this queue.	null	MEDIUM
camel.sink.endpoint.policy	The policy for this queue	null	MEDIUM
camel.sink.endpoint.receiveMessageWaitTimeSeconds	If you do not specify WaitTimeSeconds in the request, the queue attribute ReceiveMessageWaitTimeSeconds is used to determine how long to wait.	null	MEDIUM
camel.sink.endpoint.redrivePolicy	Specify the policy that send message to DeadLetter queue. See detail at Amazon docs.	null	MEDIUM
camel.sink.endpoint.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.sink.endpoint.secretKey	Amazon AWS Secret Key	null	MEDIUM
camel.component.aws2-sqs.amazonAWSHost	The hostname of the Amazon AWS cloud.	"amazonaws.com"	MEDIUM
camel.component.aws2-sqs.amazonSQSClient	To use the AmazonSQS as client	null	MEDIUM
camel.component.aws2-sqs.autoCreateQueue	Setting the autocreation of the queue	true	MEDIUM
camel.component.aws2-sqs.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-sqs.configuration	The AWS SQS default configuration	null	MEDIUM
camel.component.aws2-sqs.protocol	The underlying protocol used to communicate with SQS	"https"	MEDIUM
camel.component.aws2-sqs.proxyProtocol	To define a proxy protocol when instantiating the SQS client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.component.aws2-sqs.queueOwnerAWSAccountId	Specify the queue owner aws account id when you need to connect the queue with different account owner.	null	MEDIUM
camel.component.aws2-sqs.region	The region in which SQS client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.component.aws2-sqs.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.component.aws2-sqs.delaySeconds	Delay sending messages for a number of seconds.	null	MEDIUM
camel.component.aws2-sqs.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.aws2-sqs.messageDeduplicationId Strategy	Only for FIFO queues. Strategy for setting the messageDeduplicationId on the message. Can be one of the following options: useExchangeId, useContentBasedDeduplication. For the useContentBasedDeduplication option, no messageDeduplicationId will be set on the message. One of: [useExchangeId] [useContentBasedDeduplication]	"useExchangeId"	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-sqs.messageGroupIdStrategy	Only for FIFO queues. Strategy for setting the messageGroupId on the message. Can be one of the following options: useConstant, useExchangeId, usePropertyValue. For the usePropertyValue option, the value of property CamelAwsMessageGroupId will be used. One of: [useConstant] [useExchangeId] [usePropertyValue]	null	MEDIUM
camel.component.aws2-sqs.operation	The operation to do in case the user don't want to send only a message One of: [sendBatchMessage] [deleteMessage] [listQueues] [purgeQueue]	null	MEDIUM
camel.component.aws2-sqs.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.aws2-sqs.delayQueue	Define if you want to apply delaySeconds option to the queue or on single messages	false	MEDIUM
camel.component.aws2-sqs.queueUrl	To define the queueUrl explicitly. All other parameters, which would influence the queueUrl, are ignored. This parameter is intended to be used, to connect to a mock implementation of SQS, for testing purposes.	null	MEDIUM
camel.component.aws2-sqs.proxyHost	To define a proxy host when instantiating the SQS client	null	MEDIUM
camel.component.aws2-sqs.proxyPort	To define a proxy port when instantiating the SQS client	null	MEDIUM
camel.component.aws2-sqs.maximumMessageSize	The maximumMessageSize (in bytes) an SQS message can contain for this queue.	null	MEDIUM
camel.component.aws2-sqs.messageRetentionPeriod	The messageRetentionPeriod (in seconds) a message will be retained by SQS for this queue.	null	MEDIUM
camel.component.aws2-sqs.policy	The policy for this queue	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-sqs.receiveMessageWaitTime Seconds	If you do not specify WaitTimeSeconds in the request, the queue attribute ReceiveMessageWaitTimeSeconds is used to determine how long to wait.	null	MEDIUM
camel.component.aws2-sqs.redrivePolicy	Specify the policy that send message to DeadLetter queue. See detail at Amazon docs.	null	MEDIUM
camel.component.aws2-sqs.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.component.aws2-sqs.secretKey	Amazon AWS Secret Key	null	MEDIUM

The camel-aws2-sqs sink connector has no converters out of the box.

The camel-aws2-sqs sink connector has no transforms out of the box.

The camel-aws2-sqs sink connector has no aggregation strategies out of the box.

5.7. CAMEL-AWS2-SQS-KAFKA-CONNECTOR SOURCE CONFIGURATION

When using camel-aws2-sqs-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-aws2-sqs-kafka-connector</artifactId>
  <version>x.x.x</version>
  <!-- use the same version as your Camel Kafka connector version -->
</dependency>
```

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.aws2sqs.CamelAws2sqsSourceConnector
```

The camel-aws2-sqs source connector supports 91 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.queueNameOrArn	Queue name or ARN	null	HIGH
camel.source.endpoint.amazonAWSHost	The hostname of the Amazon AWS cloud.	"amazonaws.com"	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.amazonSQSClient	To use the AmazonSQS as client	null	MEDIUM
camel.source.endpoint.autoCreateQueue	Setting the autocreation of the queue	true	MEDIUM
camel.source.endpoint.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM
camel.source.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to map headers to/from Camel.	null	MEDIUM
camel.source.endpoint.protocol	The underlying protocol used to communicate with SQS	"https"	MEDIUM
camel.source.endpoint.proxyProtocol	To define a proxy protocol when instantiating the SQS client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.source.endpoint.queueOwnerAWSAccountId	Specify the queue owner aws account id when you need to connect the queue with different account owner.	null	MEDIUM
camel.source.endpoint.region	The region in which SQS client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.source.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.source.endpoint.attributeNames	A list of attribute names to receive when consuming. Multiple names can be separated by comma.	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.concurrentConsumers	Allows you to use multiple threads to poll the sqs queue to increase throughput	1	MEDIUM
camel.source.endpoint.defaultVisibilityTimeout	The default visibility timeout (in seconds)	null	MEDIUM
camel.source.endpoint.deleteAfterRead	Delete message from SQS after it has been read	true	MEDIUM
camel.source.endpoint.deleteIfFiltered	Whether or not to send the DeleteMessage to the SQS queue if an exchange fails to get through a filter. If 'false' and exchange does not make it through a Camel filter upstream in the route, then don't send DeleteMessage.	true	MEDIUM
camel.source.endpoint.extendMessageVisibility	If enabled then a scheduled background task will keep extending the message visibility on SQS. This is needed if it takes a long time to process the message. If set to true defaultVisibilityTimeout must be set. See details at Amazon docs.	false	MEDIUM
camel.source.endpoint.kmsDataKeyReusePeriodSeconds	The length of time, in seconds, for which Amazon SQS can reuse a data key to encrypt or decrypt messages before calling AWS KMS again. An integer representing seconds, between 60 seconds (1 minute) and 86,400 seconds (24 hours). Default: 300 (5 minutes).	null	MEDIUM
camel.source.endpoint.kmsMasterKeyId	The ID of an AWS-managed customer master key (CMK) for Amazon SQS or a custom CMK.	null	MEDIUM
camel.source.endpoint.maxMessagesPerPoll	Gets the maximum number of messages as a limit to poll at each polling. Is default unlimited, but use 0 or negative number to disable it as unlimited.	null	MEDIUM
camel.source.endpoint.messageAttributeNames	A list of message attribute names to receive when consuming. Multiple names can be separated by comma.	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.serverSideEncryptionEnabled	Define if Server Side Encryption is enabled or not on the queue	false	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.visibilityTimeout	The duration (in seconds) that the received messages are hidden from subsequent retrieve requests after being retrieved by a ReceiveMessage request to set in the com.amazonaws.services.sqs.model.SetQueueAttributesRequest. This only make sense if its different from defaultVisibilityTimeout. It changes the queue visibility timeout attribute permanently.	null	MEDIUM
camel.source.endpoint.waitTimeSeconds	Duration in seconds (0 to 20) that the ReceiveMessage action call will wait until a message is in the queue to include in the response.	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.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.delayQueue	Define if you want to apply delaySeconds option to the queue or on single messages	false	MEDIUM
camel.source.endpoint.queueUrl	To define the queueUrl explicitly. All other parameters, which would influence the queueUrl, are ignored. This parameter is intended to be used, to connect to a mock implementation of SQS, for testing purposes.	null	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

Name	Description	Default	Priority
camel.source.endpoint.proxyHost	To define a proxy host when instantiating the SQS client	null	MEDIUM
camel.source.endpoint.proxyPort	To define a proxy port when instantiating the SQS client	null	MEDIUM
camel.source.endpoint.maximumMessageSize	The maximumMessageSize (in bytes) an SQS message can contain for this queue.	null	MEDIUM
camel.source.endpoint.messageRetentionPeriod	The messageRetentionPeriod (in seconds) a message will be retained by SQS for this queue.	null	MEDIUM
camel.source.endpoint.policy	The policy for this queue	null	MEDIUM
camel.source.endpoint.receiveMessageWaitTime Seconds	If you do not specify WaitTimeSeconds in the request, the queue attribute ReceiveMessageWaitTimeSeconds is used to determine how long to wait.	null	MEDIUM
camel.source.endpoint.redrivePolicy	Specify the policy that send message to DeadLetter queue. See detail at Amazon docs.	null	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

Name	Description	Default	Priority
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. Use value spring or quartz for built in scheduler	"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

Name	Description	Default	Priority
camel.component.aws2-sqs.amazonAWSHost	The hostname of the Amazon AWS cloud.	"amazonaws.com"	MEDIUM
camel.component.aws2-sqs.amazonSQSClient	To use the AmazonSQS as client	null	MEDIUM
camel.component.aws2-sqs.autoCreateQueue	Setting the autocreation of the queue	true	MEDIUM
camel.component.aws2-sqs.autoDiscoverClient	Setting the autoDiscoverClient mechanism, if true, the component will look for a client instance in the registry automatically otherwise it will skip that checking.	true	MEDIUM
camel.component.aws2-sqs.configuration	The AWS SQS default configuration	null	MEDIUM
camel.component.aws2-sqs.protocol	The underlying protocol used to communicate with SQS	"https"	MEDIUM
camel.component.aws2-sqs.proxyProtocol	To define a proxy protocol when instantiating the SQS client One of: [HTTP] [HTTPS]	"HTTPS"	MEDIUM
camel.component.aws2-sqs.queueOwnerAWSAccountId	Specify the queue owner aws account id when you need to connect the queue with different account owner.	null	MEDIUM
camel.component.aws2-sqs.region	The region in which SQS client needs to work. When using this parameter, the configuration will expect the lowercase name of the region (for example ap-east-1) You'll need to use the name Region.EU_WEST_1.id()	null	MEDIUM
camel.component.aws2-sqs.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	MEDIUM
camel.component.aws2-sqs.attributeNames	A list of attribute names to receive when consuming. Multiple names can be separated by comma.	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-sqs.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.aws2-sqs.concurrentConsumers	Allows you to use multiple threads to poll the sqs queue to increase throughput	1	MEDIUM
camel.component.aws2-sqs.defaultVisibilityTimeout	The default visibility timeout (in seconds)	null	MEDIUM
camel.component.aws2-sqs.deleteAfterRead	Delete message from SQS after it has been read	true	MEDIUM
camel.component.aws2-sqs.deleteIfFiltered	Whether or not to send the DeleteMessage to the SQS queue if an exchange fails to get through a filter. If 'false' and exchange does not make it through a Camel filter upstream in the route, then don’t send DeleteMessage.	true	MEDIUM
camel.component.aws2-sqs.extendMessageVisibility	If enabled then a scheduled background task will keep extending the message visibility on SQS. This is needed if it takes a long time to process the message. If set to true defaultVisibilityTimeout must be set. See details at Amazon docs.	false	MEDIUM
camel.component.aws2-sqs.kmsDataKeyReusePeriod Seconds	The length of time, in seconds, for which Amazon SQS can reuse a data key to encrypt or decrypt messages before calling AWS KMS again. An integer representing seconds, between 60 seconds (1 minute) and 86,400 seconds (24 hours). Default: 300 (5 minutes).	null	MEDIUM
camel.component.aws2-sqs.kmsMasterKeyId	The ID of an AWS-managed customer master key (CMK) for Amazon SQS or a custom CMK.	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-sqs.messageAttributeNames	A list of message attribute names to receive when consuming. Multiple names can be separated by comma.	null	MEDIUM
camel.component.aws2-sqs.serverSideEncryption Enabled	Define if Server Side Encryption is enabled or not on the queue	false	MEDIUM
camel.component.aws2-sqs.visibilityTimeout	The duration (in seconds) that the received messages are hidden from subsequent retrieve requests after being retrieved by a ReceiveMessage request to set in the com.amazonaws.services.sqs.model.SetQueue AttributesRequest. This only make sense if its different from defaultVisibilityTimeout. It changes the queue visibility timeout attribute permanently.	null	MEDIUM
camel.component.aws2-sqs.waitTimeSeconds	Duration in seconds (0 to 20) that the ReceiveMessage action call will wait until a message is in the queue to include in the response.	null	MEDIUM
camel.component.aws2-sqs.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.aws2-sqs.delayQueue	Define if you want to apply delaySeconds option to the queue or on single messages	false	MEDIUM
camel.component.aws2-sqs.queueUrl	To define the queueUrl explicitly. All other parameters, which would influence the queueUrl, are ignored. This parameter is intended to be used, to connect to a mock implementation of SQS, for testing purposes.	null	MEDIUM
camel.component.aws2-sqs.proxyHost	To define a proxy host when instantiating the SQS client	null	MEDIUM
camel.component.aws2-sqs.proxyPort	To define a proxy port when instantiating the SQS client	null	MEDIUM
camel.component.aws2-sqs.maximumMessageSize	The maximumMessageSize (in bytes) an SQS message can contain for this queue.	null	MEDIUM

Name	Description	Default	Priority
camel.component.aws2-sqs.messageRetentionPeriod	The messageRetentionPeriod (in seconds) a message will be retained by SQS for this queue.	null	MEDIUM
camel.component.aws2-sqs.policy	The policy for this queue	null	MEDIUM
camel.component.aws2-sqs.receiveMessageWaitTime Seconds	If you do not specify WaitTimeSeconds in the request, the queue attribute ReceiveMessageWaitTimeSeconds is used to determine how long to wait.	null	MEDIUM
camel.component.aws2-sqs.redrivePolicy	Specify the policy that send message to DeadLetter queue. See detail at Amazon docs.	null	MEDIUM
camel.component.aws2-sqs.accessKey	Amazon AWS Access Key	null	MEDIUM
camel.component.aws2-sqs.secretKey	Amazon AWS Secret Key	null	MEDIUM

The camel-aws2-sqs sink connector has no converters out of the box.

The camel-aws2-sqs sink connector has no transforms out of the box.

The camel-aws2-sqs sink connector has no aggregation strategies out of the box.

5.8. 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>
```

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.cql.CamelCqlSinkConnector
```

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
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.clusterName	Cluster name	null	MEDIUM
camel.sink.endpoint.consistencyLevel	Consistency level to use One of: [ANY] [ONE] [TWO] [THREE] [QUORUM] [ALL] [LOCAL_ONE] [LOCAL_QUORUM] [EACH_QUORUM] [SERIAL] [LOCAL_SERIAL]	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.datacenter	Datacenter to use	"datacenter1"	MEDIUM
camel.sink.endpoint.loadBalancingPolicyClass	To use a specific LoadBalancingPolicyClass	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

The camel-cql sink connector has no converters out of the box.

The camel-cql sink connector has no transforms out of the box.

The camel-cql sink connector has no aggregation strategies out of the box.

5.9. CAMEL-CQL-KAFKA-CONNECTOR SOURCE CONFIGURATION

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

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.cql.CamelCqlSourceConnector
```

The camel-cql source connector supports 37 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.beanRef	beanRef is defined using bean:id	null	MEDIUM
camel.source.path.hosts	Hostname(s) cassansdra server(s). Multiple hosts can be separated by comma.	null	MEDIUM
camel.source.path.port	Port number of cassansdra server(s)	null	MEDIUM
camel.source.path.keyspace	Keyspace to use	null	MEDIUM
camel.source.endpoint.clusterName	Cluster name	null	MEDIUM
camel.source.endpoint.consistencyLevel	Consistency level to use One of: [ANY] [ONE] [TWO] [THREE] [QUORUM] [ALL] [LOCAL_ONE] [LOCAL_QUORUM] [EACH_QUORUM] [SERIAL] [LOCAL_SERIAL]	null	MEDIUM
camel.source.endpoint.cql	CQL query to perform. Can be overridden with the message header with key CamelCqlQuery.	null	MEDIUM
camel.source.endpoint.datacenter	Datacenter to use	"datacenter1"	MEDIUM
camel.source.endpoint.loadBalancingPolicyClass	To use a specific LoadBalancingPolicyClass	null	MEDIUM
camel.source.endpoint.password	Password for session authentication	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.prepareStatements	Whether to use PreparedStatements or regular Statements	true	MEDIUM
camel.source.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.source.endpoint.session	To use the Session instance (you would normally not use this option)	null	MEDIUM
camel.source.endpoint.username	Username for session authentication	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
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.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.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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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. Use value spring or quartz for built in scheduler	"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.component.cql.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.cql.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM

The camel-cql sink connector has no converters out of the box.

The camel-cql sink connector has no transforms out of the box.

The camel-cql sink connector has no aggregation strategies out of the box.

5.10. 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>
```

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.elasticsearchrest.CamelElasticsearchrestSinkConnector
```

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

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.maxRetryTimeout	The time in ms before retry	30000	MEDIUM
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

Name	Description	Default	Priority
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.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

The camel-elasticsearch-rest sink connector has no converters out of the box.

The camel-elasticsearch-rest sink connector has no transforms out of the box.

The camel-elasticsearch-rest sink connector has no aggregation strategies out of the box.

5.11. CAMEL-FILE-KAFKA-CONNECTOR SINK CONFIGURATION

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

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.file.CamelFileSinkConnector
```

The camel-file sink connector supports 27 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.directoryName	The starting directory	null	HIGH

Name	Description	Default	Priority
camel.sink.endpoint.charset	This option is used to specify the encoding of the file. You can use this on the consumer, to specify the encodings of the files, which allow Camel to know the charset it should load the file content in case the file content is being accessed. Likewise when writing a file, you can use this option to specify which charset to write the file as well. Do mind that when writing the file Camel may have to read the message content into memory to be able to convert the data into the configured charset, so do not use this if you have big messages.	null	MEDIUM
camel.sink.endpoint.doneFileName	Producer: If provided, then Camel will write a 2nd done file when the original file has been written. The done file will be empty. This option configures what file name to use. Either you can specify a fixed name. Or you can use dynamic placeholders. The done file will always be written in the same folder as the original file. Consumer: If provided, Camel will only consume files if a done file exists. This option configures what file name to use. Either you can specify a fixed name. Or you can use dynamic placeholders.The done file is always expected in the same folder as the original file. Only \${file.name} and \${file.name.next} is supported as dynamic placeholders.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.fileName	Use Expression such as File Language to dynamically set the filename. For consumers, it's used as a filename filter. For producers, it's used to evaluate the filename to write. If an expression is set, it take precedence over the CamelFileName header. (Note: The header itself can also be an Expression). The expression options support both String and Expression types. If the expression is a String type, it is always evaluated using the File Language. If the expression is an Expression type, the specified Expression type is used - this allows you, for instance, to use OGNL expressions. For the consumer, you can use it to filter filenames, so you can for instance consume today's file using the File Language syntax: mydata-\${date:now:yyyyMMdd}.txt. The producers support the CamelOverruleFileName header which takes precedence over any existing CamelFileName header; the CamelOverruleFileName is a header that is used only once, and makes it easier as this avoids to temporary store CamelFileName and have to restore it afterwards.	null	MEDIUM
camel.sink.endpoint.appendChars	Used to append characters (text) after writing files. This can for example be used to add new lines or other separators when writing and appending to existing files. To specify new-line (slash-n or slash-r) or tab (slash-t) characters then escape with an extra slash, eg slash-slash-n.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.fileExist	What to do if a file already exists with the same name. Override, which is the default, replaces the existing file. - Append - adds content to the existing file. - Fail - throws a GenericFileOperationException, indicating that there is already an existing file. - Ignore - silently ignores the problem and does not override the existing file, but assumes everything is okay. - Move - option requires to use the moveExisting option to be configured as well. The option eagerDeleteTargetFile can be used to control what to do if an moving the file, and there exists already an existing file, otherwise causing the move operation to fail. The Move option will move any existing files, before writing the target file. - TryRename is only applicable if tempFileName option is in use. This allows to try renaming the file from the temporary name to the actual name, without doing any exists check. This check may be faster on some file systems and especially FTP servers. One of: [Override] [Append] [Fail] [Ignore] [Move] [TryRename]	"Override"	MEDIUM
camel.sink.endpoint.flatten	Flatten is used to flatten the file name path to strip any leading paths, so it's just the file name. This allows you to consume recursively into sub-directories, but when you eg write the files to another directory they will be written in a single directory. Setting this to true on the producer enforces that any file name in CamelFileName header will be stripped for any leading paths.	false	MEDIUM
camel.sink.endpoint.jailStartingDirectory	Used for jailing (restricting) writing files to the starting directory (and sub) only. This is enabled by default to not allow Camel to write files to outside directories (to be more secured out of the box). You can turn this off to allow writing files to directories outside the starting directory, such as parent or root folders.	true	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.moveExisting	Expression (such as File Language) used to compute file name to use when fileExist=Move is configured. To move files into a backup subdirectory just enter backup. This option only supports the following File Language tokens: file:name, file:name.ext, file:name.noext, file:onlyname, file:onlyname.noext, file:ext, and file:parent. Notice the file:parent is not supported by the FTP component, as the FTP component can only move any existing files to a relative directory based on current dir as base.	null	MEDIUM
camel.sink.endpoint.tempFileName	The same as tempPrefix option but offering a more fine grained control on the naming of the temporary filename as it uses the File Language. The location for tempFilename is relative to the final file location in the option 'fileName', not the target directory in the base uri. For example if option fileName includes a directory prefix: dir/finalFilename then tempFileName is relative to that subdirectory dir.	null	MEDIUM
camel.sink.endpoint.tempPrefix	This option is used to write the file using a temporary name and then, after the write is complete, rename it to the real name. Can be used to identify files being written and also avoid consumers (not using exclusive read locks) reading in progress files. Is often used by FTP when uploading big files.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.allowNullBody	Used to specify if a null body is allowed during file writing. If set to true then an empty file will be created, when set to false, and attempting to send a null body to the file component, a GenericFileWriteException of 'Cannot write null body to file.' will be thrown. If the fileExist option is set to 'Override', then the file will be truncated, and if set to append the file will remain unchanged.	false	MEDIUM
camel.sink.endpoint.chmod	Specify the file permissions which is sent by the producer, the chmod value must be between 000 and 777; If there is a leading digit like in 0755 we will ignore it.	null	MEDIUM
camel.sink.endpoint.chmodDirectory	Specify the directory permissions used when the producer creates missing directories, the chmod value must be between 000 and 777; If there is a leading digit like in 0755 we will ignore it.	null	MEDIUM
camel.sink.endpoint.eagerDeleteTargetFile	Whether or not to eagerly delete any existing target file. This option only applies when you use fileExists=Override and the tempFileName option as well. You can use this to disable (set it to false) deleting the target file before the temp file is written. For example you may write big files and want the target file to exists during the temp file is being written. This ensure the target file is only deleted until the very last moment, just before the temp file is being renamed to the target filename. This option is also used to control whether to delete any existing files when fileExist=Move is enabled, and an existing file exists. If this option copyAndDeleteOnRenameFails false, then an exception will be thrown if an existing file existed, if its true, then the existing file is deleted before the move operation.	true	MEDIUM
camel.sink.endpoint.forceWrites	Whether to force syncing writes to the file system. You can turn this off if you do not want this level of guarantee, for example if writing to logs / audit logs etc; this would yield better performance.	true	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.keepLastModified	Will keep the last modified timestamp from the source file (if any). Will use the Exchange.FILE_LAST_MODIFIED header to located the timestamp. This header can contain either a java.util.Date or long with the timestamp. If the timestamp exists and the option is enabled it will set this timestamp on the written file. Note: This option only applies to the file producer. You cannot use this option with any of the \ftp producers.	false	MEDIUM
camel.sink.endpoint.moveExistingFileStrategy	Strategy (Custom Strategy) used to move file with special naming token to use when fileExist=Move is configured. By default, there is an implementation used if no custom strategy is provided	null	MEDIUM
camel.sink.endpoint.autoCreate	Automatically create missing directories in the file's pathname. For the file consumer, that means creating the starting directory. For the file producer, it means the directory the files should be written to.	true	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.bufferSize	Buffer size in bytes used for writing files (or in case of FTP for downloading and uploading files).	131072	MEDIUM
camel.sink.endpoint.copyAndDeleteOnRenameFail	Whether to fallback and do a copy and delete file, in case the file could not be renamed directly. This option is not available for the FTP component.	true	MEDIUM
camel.sink.endpoint.renameUsingCopy	Perform rename operations using a copy and delete strategy. This is primarily used in environments where the regular rename operation is unreliable (e.g. across different file systems or networks). This option takes precedence over the copyAndDeleteOnRenameFail parameter that will automatically fall back to the copy and delete strategy, but only after additional delays.	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.file.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.file.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM

The camel-file sink connector has no converters out of the box.

The camel-file sink connector has no transforms out of the box.

The camel-file sink connector has no aggregation strategies out of the box.

5.12. CAMEL-HDFS-KAFKA-CONNECTOR SINK CONFIGURATION

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

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.hdfs.CamelHdfsSinkConnector
```

The camel-hdfs sink connector supports 32 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.hostName	HDFS host to use	null	HIGH
camel.sink.path.port	HDFS port to use	8020	MEDIUM

Name	Description	Default	Priority
camel.sink.path.path	The directory path to use	null	HIGH
camel.sink.endpoint.connectOnStartup	Whether to connect to the HDFS file system on starting the producer/consumer. If false then the connection is created on-demand. Notice that HDFS may take up till 15 minutes to establish a connection, as it has hardcoded 45 x 20 sec redelivery. By setting this option to false allows your application to startup, and not block for up till 15 minutes.	true	MEDIUM
camel.sink.endpoint.fileSystemType	Set to LOCAL to not use HDFS but local java.io.File instead. One of: [LOCAL] [HDFS]	"HDFS"	MEDIUM
camel.sink.endpoint.fileType	The file type to use. For more details see Hadoop HDFS documentation about the various files types. One of: [NORMAL_FILE] [SEQUENCE_FILE] [MAP_FILE] [BLOOMMAP_FILE] [ARRAY_FILE]	"NORMAL_FILE"	MEDIUM
camel.sink.endpoint.keyType	The type for the key in case of sequence or map files. One of: [NULL] [BOOLEAN] [BYTE] [INT] [FLOAT] [LONG] [DOUBLE] [TEXT] [BYTES]	"NULL"	MEDIUM
camel.sink.endpoint.namedNodes	A comma separated list of named nodes (e.g. srv11.example.com:8020,srv12.example.com:8020)	null	MEDIUM
camel.sink.endpoint.owner	The file owner must match this owner for the consumer to pickup the file. Otherwise the file is skipped.	null	MEDIUM
camel.sink.endpoint.valueType	The type for the key in case of sequence or map files One of: [NULL] [BOOLEAN] [BYTE] [INT] [FLOAT] [LONG] [DOUBLE] [TEXT] [BYTES]	"BYTES"	MEDIUM
camel.sink.endpoint.append	Append to existing file. Notice that not all HDFS file systems support the append option.	false	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.overwrite	Whether to overwrite existing files with the same name	true	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.blockSize	The size of the HDFS blocks	67108864L	MEDIUM
camel.sink.endpoint.bufferSize	The buffer size used by HDFS	4096	MEDIUM
camel.sink.endpoint.checkIdleInterval	How often (time in millis) in to run the idle checker background task. This option is only in use if the splitter strategy is IDLE.	500	MEDIUM
camel.sink.endpoint.chunkSize	When reading a normal file, this is split into chunks producing a message per chunk.	4096	MEDIUM
camel.sink.endpoint.compressionCodec	The compression codec to use One of: [DEFAULT] [GZIP] [BZIP2]	"DEFAULT"	MEDIUM
camel.sink.endpoint.compressionType	The compression type to use (is default not in use) One of: [NONE] [RECORD] [BLOCK]	"NONE"	MEDIUM
camel.sink.endpoint.openedSuffix	When a file is opened for reading/writing the file is renamed with this suffix to avoid to read it during the writing phase.	"opened"	MEDIUM
camel.sink.endpoint.readSuffix	Once the file has been read is renamed with this suffix to avoid to read it again.	"read"	MEDIUM
camel.sink.endpoint.replication	The HDFS replication factor	3	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.splitStrategy	In the current version of Hadoop opening a file in append mode is disabled since it's not very reliable. So, for the moment, it's only possible to create new files. The Camel HDFS endpoint tries to solve this problem in this way: If the split strategy option has been defined, the hdfs path will be used as a directory and files will be created using the configured UuidGenerator. Every time a splitting condition is met, a new file is created. The splitStrategy option is defined as a string with the following syntax: splitStrategy=ST:value,ST:value,... where ST can be: BYTES a new file is created, and the old is closed when the number of written bytes is more than value MESSAGES a new file is created, and the old is closed when the number of written messages is more than value IDLE a new file is created, and the old is closed when no writing happened in the last value milliseconds	null	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.kerberosConfigFileLocation	The location of the kerb5.conf file (https://web.mit.edu/kerberos/krb5-1.12/doc/admin/conf_files/krb5_conf.html)	null	MEDIUM
camel.sink.endpoint.kerberosKeytabLocation	The location of the keytab file used to authenticate with the kerberos nodes (contains pairs of kerberos principals and encrypted keys (which are derived from the Kerberos password))	null	MEDIUM
camel.sink.endpoint.kerberosUsername	The username used to authenticate with the kerberos nodes	null	MEDIUM

Name	Description	Default	Priority
camel.component.hdfs.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.hdfs.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.hdfs.jAASConfiguration	To use the given configuration for security with JAAS.	null	MEDIUM
camel.component.hdfs.kerberosConfigFile	To use kerberos authentication, set the value of the 'java.security.krb5.conf' environment variable to an existing file. If the environment variable is already set, warn if different than the specified parameter	null	MEDIUM

The camel-hdfs sink connector has no converters out of the box.

The camel-hdfs sink connector has no transforms out of the box.

The camel-hdfs sink connector has no aggregation strategies out of the box.

5.13. CAMEL-HTTP-KAFKA-CONNECTOR SINK CONFIGURATION

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

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.http.CamelHttpSinkConnector
```

The camel-http sink connector supports 80 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.httpUri	The url of the HTTP endpoint to call.	null	HIGH
camel.sink.endpoint.disableStreamCache	Determines whether or not the raw input stream from Servlet is cached or not (Camel will read the stream into a in memory/overflow to file, Stream caching) cache. By default Camel will cache the Servlet input stream to support reading it multiple times to ensure it Camel can retrieve all data from the stream. However you can set this option to true when you for example need to access the raw stream, such as streaming it directly to a file or other persistent store. DefaultHttpBinding will copy the request input stream into a stream cache and put it into message body if this option is false to support reading the stream multiple times. If you use Servlet to bridge/proxy an endpoint then consider enabling this option to improve performance, in case you do not need to read the message payload multiple times. The http producer will by default cache the response body stream. If setting this option to true, then the producers will not cache the response body stream but use the response stream as-is as the message body.	false	MEDIUM
camel.sink.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to filter header to and from Camel message.	null	MEDIUM
camel.sink.endpoint.httpBinding	To use a custom HttpBinding to control the mapping between Camel message and HttpClient.	null	MEDIUM
camel.sink.endpoint.bridgeEndpoint	If the option is true, HttpProducer will ignore the Exchange.HTTP_URI header, and use the endpoint's URI for request. You may also set the option throwExceptionOnFailure to be false to let the HttpProducer send all the fault response back.	false	MEDIUM
camel.sink.endpoint.chunked	If this option is false the Servlet will disable the HTTP streaming and set the content-length header on the response	true	MEDIUM
camel.sink.endpoint.clearExpiredCookies	Whether to clear expired cookies before sending the HTTP request. This ensures the cookies store does not keep growing by adding new cookies which is newer removed when they are expired.	true	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.connectionClose	Specifies whether a Connection Close header must be added to HTTP Request. By default connectionClose is false.	false	MEDIUM
camel.sink.endpoint.copyHeaders	If this option is true then IN exchange headers will be copied to OUT exchange headers according to copy strategy. Setting this to false, allows to only include the headers from the HTTP response (not propagating IN headers).	true	MEDIUM
camel.sink.endpoint.customHostHeader	To use custom host header for producer. When not set in query will be ignored. When set will override host header derived from url.	null	MEDIUM
camel.sink.endpoint.httpMethod	Configure the HTTP method to use. The HttpMethod header cannot override this option if set. One of: [GET] [POST] [PUT] [DELETE] [HEAD] [OPTIONS] [TRACE] [PATCH]	null	MEDIUM
camel.sink.endpoint.ignoreResponseBody	If this option is true, The http producer won't read response body and cache the input stream	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.preserveHostHeader	If the option is true, HttpProducer will set the Host header to the value contained in the current exchange Host header, useful in reverse proxy applications where you want the Host header received by the downstream server to reflect the URL called by the upstream client, this allows applications which use the Host header to generate accurate URL's for a proxied service	false	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.throwExceptionOnFailure	Option to disable throwing the HttpOperationFailedException in case of failed responses from the remote server. This allows you to get all responses regardless of the HTTP status code.	true	MEDIUM
camel.sink.endpoint.transferException	If enabled and an Exchange failed processing on the consumer side, and if the caused Exception was send back serialized in the response as a application/x-java-serialized-object content type. On the producer side the exception will be deserialized and thrown as is, instead of the HttpOperationFailedException. The caused exception is required to be serialized. This is by default turned off. If you enable this then be aware that Java will deserialize the incoming data from the request to Java and that can be a potential security risk.	false	MEDIUM
camel.sink.endpoint.cookieHandler	Configure a cookie handler to maintain a HTTP session	null	MEDIUM
camel.sink.endpoint.cookieStore	To use a custom CookieStore. By default the BasicCookieStore is used which is an in-memory only cookie store. Notice if bridgeEndpoint=true then the cookie store is forced to be a noop cookie store as cookie shouldn't be stored as we are just bridging (eg acting as a proxy). If a cookieHandler is set then the cookie store is also forced to be a noop cookie store as cookie handling is then performed by the cookieHandler.	null	MEDIUM
camel.sink.endpoint.deleteWithBody	Whether the HTTP DELETE should include the message body or not. By default HTTP DELETE do not include any HTTP body. However in some rare cases users may need to be able to include the message body.	false	MEDIUM
camel.sink.endpoint.getWithBody	Whether the HTTP GET should include the message body or not. By default HTTP GET do not include any HTTP body. However in some rare cases users may need to be able to include the message body.	false	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.okStatusCodeRange	The status codes which are considered a success response. The values are inclusive. Multiple ranges can be defined, separated by comma, e.g. 200-204,209,301-304. Each range must be a single number or from-to with the dash included.	"200-299"	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.clientBuilder	Provide access to the http client request parameters used on new RequestConfig instances used by producers or consumers of this endpoint.	null	MEDIUM
camel.sink.endpoint.clientConnectionManager	To use a custom HttpClientConnectionManager to manage connections	null	MEDIUM
camel.sink.endpoint.connectionsPerRoute	The maximum number of connections per route.	20	MEDIUM
camel.sink.endpoint.httpClient	Sets a custom HttpClient to be used by the producer	null	MEDIUM
camel.sink.endpoint.httpClientConfigurer	Register a custom configuration strategy for new HttpClient instances created by producers or consumers such as to configure authentication mechanisms etc.	null	MEDIUM
camel.sink.endpoint.httpClientOptions	To configure the HttpClient using the key/values from the Map.	null	MEDIUM
camel.sink.endpoint.httpContext	To use a custom HttpContext instance	null	MEDIUM
camel.sink.endpoint.mapHttpMessageBody	If this option is true then IN exchange Body of the exchange will be mapped to HTTP body. Setting this to false will avoid the HTTP mapping.	true	MEDIUM
camel.sink.endpoint.mapHttpMessageFormUrlEncoded Body	If this option is true then IN exchange Form Encoded body of the exchange will be mapped to HTTP. Setting this to false will avoid the HTTP Form Encoded body mapping.	true	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.mapHttpMessageHeaders	If this option is true then IN exchange Headers of the exchange will be mapped to HTTP headers. Setting this to false will avoid the HTTP Headers mapping.	true	MEDIUM
camel.sink.endpoint.maxTotalConnections	The maximum number of connections.	200	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.useSystemProperties	To use System Properties as fallback for configuration	false	MEDIUM
camel.sink.endpoint.proxyAuthDomain	Proxy authentication domain to use with NTML	null	MEDIUM
camel.sink.endpoint.proxyAuthHost	Proxy authentication host	null	MEDIUM
camel.sink.endpoint.proxyAuthMethod	Proxy authentication method to use One of: [Basic] [Digest] [NTLM]	null	MEDIUM
camel.sink.endpoint.proxyAuthNtHost	Proxy authentication domain (workstation name) to use with NTML	null	MEDIUM
camel.sink.endpoint.proxyAuthPassword	Proxy authentication password	null	MEDIUM
camel.sink.endpoint.proxyAuthPort	Proxy authentication port	null	MEDIUM
camel.sink.endpoint.proxyAuthScheme	Proxy authentication scheme to use One of: [http] [https]	null	MEDIUM
camel.sink.endpoint.proxyAuthUsername	Proxy authentication username	null	MEDIUM
camel.sink.endpoint.proxyHost	Proxy hostname to use	null	MEDIUM
camel.sink.endpoint.proxyPort	Proxy port to use	null	MEDIUM
camel.sink.endpoint.authDomain	Authentication domain to use with NTML	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.authenticationPreemptive	If this option is true, camel-http sends preemptive basic authentication to the server.	false	MEDIUM
camel.sink.endpoint.authHost	Authentication host to use with NTML	null	MEDIUM
camel.sink.endpoint.authMethod	Authentication methods allowed to use as a comma separated list of values Basic, Digest or NTLM.	null	MEDIUM
camel.sink.endpoint.authMethodPriority	Which authentication method to prioritize to use, either as Basic, Digest or NTLM. One of: [Basic] [Digest] [NTLM]	null	MEDIUM
camel.sink.endpoint.authPassword	Authentication password	null	MEDIUM
camel.sink.endpoint.authUsername	Authentication username	null	MEDIUM
camel.sink.endpoint.sslContextParameters	To configure security using SSLContextParameters. Important: Only one instance of org.apache.camel.util.jsse.SSLContextParameters is supported per HttpComponent. If you need to use 2 or more different instances, you need to define a new HttpComponent per instance you need.	null	MEDIUM
camel.sink.endpoint.x509HostnameVerifier	To use a custom X509HostnameVerifier such as DefaultHostnameVerifier or NoopHostnameVerifier	null	MEDIUM
camel.component.http.cookieStore	To use a custom org.apache.http.client.CookieStore. By default the org.apache.http.impl.client.BasicCookieStore is used which is an in-memory only cookie store. Notice if bridgeEndpoint=true then the cookie store is forced to be a noop cookie store as cookie shouldn't be stored as we are just bridging (eg acting as a proxy).	null	MEDIUM

Name	Description	Default	Priority
camel.component.http.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.http.allowJavaSerializedObject	Whether to allow java serialization when a request uses context-type=application/x-java-serialized-object. This is by default turned off. If you enable this then be aware that Java will deserialize the incoming data from the request to Java and that can be a potential security risk.	false	MEDIUM
camel.component.http.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.http.clientConnectionManager	To use a custom and shared HttpClientConnectionManager to manage connections. If this has been configured then this is always used for all endpoints created by this component.	null	MEDIUM
camel.component.http.connectionsPerRoute	The maximum number of connections per route.	20	MEDIUM
camel.component.http.connectionTimeToLive	The time for connection to live, the time unit is millisecond, the default value is always keep alive.	null	MEDIUM
camel.component.http.httpBinding	To use a custom HttpBinding to control the mapping between Camel message and HttpClient.	null	MEDIUM
camel.component.http.httpClientConfigurer	To use the custom HttpClientConfigurer to perform configuration of the HttpClient that will be used.	null	MEDIUM
camel.component.http.httpConfiguration	To use the shared HttpConfiguration as base configuration.	null	MEDIUM

Name	Description	Default	Priority
camel.component.http.httpContext	To use a custom org.apache.http.protocol.HttpContext when executing requests.	null	MEDIUM
camel.component.http.maxTotalConnections	The maximum number of connections.	200	MEDIUM
camel.component.http.headerFilterStrategy	To use a custom org.apache.camel.spi.HeaderFilterStrategy to filter header to and from Camel message.	null	MEDIUM
camel.component.http.proxyAuthDomain	Proxy authentication domain to use	null	MEDIUM
camel.component.http.proxyAuthHost	Proxy authentication host	null	MEDIUM
camel.component.http.proxyAuthMethod	Proxy authentication method to use One of: [Basic] [Digest] [NTLM]	null	MEDIUM
camel.component.http.proxyAuthNtHost	Proxy authentication domain (workstation name) to use with NTML	null	MEDIUM
camel.component.http.proxyAuthPassword	Proxy authentication password	null	MEDIUM
camel.component.http.proxyAuthPort	Proxy authentication port	null	MEDIUM
camel.component.http.proxyAuthUsername	Proxy authentication username	null	MEDIUM
camel.component.http.sslContextParameters	To configure security using SSLContextParameters. Important: Only one instance of org.apache.camel.support.jsse.SSLContextParameters is supported per HttpComponent. If you need to use 2 or more different instances, you need to define a new HttpComponent per instance you need.	null	MEDIUM
camel.component.http.useGlobalSslContextParameters	Enable usage of global SSL context parameters.	false	MEDIUM
camel.component.http.x509HostnameVerifier	To use a custom X509HostnameVerifier such as DefaultHostnameVerifier or NoopHostnameVerifier.	null	MEDIUM

Name	Description	Default	Priority
camel.component.http.connectionRequestTimeout	The timeout in milliseconds used when requesting a connection from the connection manager. A timeout value of zero is interpreted as an infinite timeout. A timeout value of zero is interpreted as an infinite timeout. A negative value is interpreted as undefined (system default).	-1	MEDIUM
camel.component.http.connectTimeout	Determines the timeout in milliseconds until a connection is established. A timeout value of zero is interpreted as an infinite timeout. A timeout value of zero is interpreted as an infinite timeout. A negative value is interpreted as undefined (system default).	-1	MEDIUM
camel.component.http.socketTimeout	Defines the socket timeout in milliseconds, which is the timeout for waiting for data or, put differently, a maximum period inactivity between two consecutive data packets). A timeout value of zero is interpreted as an infinite timeout. A negative value is interpreted as undefined (system default).	-1	MEDIUM

The camel-http sink connector has no converters out of the box.

The camel-http sink connector has no transforms out of the box.

The camel-http sink connector has no aggregation strategies out of the box.

5.14. CAMEL-JDBC-KAFKA-CONNECTOR SINK CONFIGURATION

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

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.jdbc.CamelJdbcSinkConnector
```

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

Name	Description	Default	Priority
camel.sink.path.dataSourceName	Name of DataSource to lookup in the Registry. If the name is dataSource or default, then Camel will attempt to lookup a default DataSource from the registry, meaning if there is a only one instance of DataSource found, then this DataSource will be used.	null	HIGH
camel.sink.endpoint.allowNamedParameters	Whether to allow using named parameters in the queries.	true	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.outputClass	Specify the full package and class name to use as conversion when outputType=SelectOne or SelectList.	null	MEDIUM
camel.sink.endpoint.outputType	Determines the output the producer should use. One of: [SelectOne] [SelectList] [StreamList]	"SelectList"	MEDIUM
camel.sink.endpoint.parameters	Optional parameters to the java.sql.Statement. For example to set maxRows, fetchSize etc.	null	MEDIUM
camel.sink.endpoint.readSize	The default maximum number of rows that can be read by a polling query. The default value is 0.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.resetAutoCommit	Camel will set the autoCommit on the JDBC connection to be false, commit the change after executed the statement and reset the autoCommit flag of the connection at the end, if the resetAutoCommit is true. If the JDBC connection doesn't support to reset the autoCommit flag, you can set the resetAutoCommit flag to be false, and Camel will not try to reset the autoCommit flag. When used with XA transactions you most likely need to set it to false so that the transaction manager is in charge of committing this tx.	true	MEDIUM
camel.sink.endpoint.transacted	Whether transactions are in use.	false	MEDIUM
camel.sink.endpoint.useGetBytesForBlob	To read BLOB columns as bytes instead of string data. This may be needed for certain databases such as Oracle where you must read BLOB columns as bytes.	false	MEDIUM
camel.sink.endpoint.useHeadersAsParameters	Set this option to true to use the prepareStatementStrategy with named parameters. This allows to define queries with named placeholders, and use headers with the dynamic values for the query placeholders.	false	MEDIUM
camel.sink.endpoint.useJDBC4ColumnNameAndLabel Semantics	Sets whether to use JDBC 4 or JDBC 3.0 or older semantic when retrieving column name. JDBC 4.0 uses columnLabel to get the column name where as JDBC 3.0 uses both columnName or columnLabel. Unfortunately JDBC drivers behave differently so you can use this option to work out issues around your JDBC driver if you get problem using this component This option is default true.	true	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.beanRowMapper	To use a custom org.apache.camel.component.jdbc.BeanRowMapper when using outputClass. The default implementation will lower case the row names and skip underscores, and dashes. For example CUST_ID is mapped as custId.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.prepareStatementStrategy	Allows the plugin to use a custom org.apache.camel.component.jdbc.JdbcPrepareStatementStrategy to control preparation of the query and prepared statement.	null	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.jdbc.dataSource	To use the DataSource instance instead of looking up the data source by name from the registry.	null	MEDIUM
camel.component.jdbc.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.jdbc.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM

The camel-jdbc sink connector has no converters out of the box.

The camel-jdbc sink connector has no transforms out of the box.

The camel-jdbc sink connector has no aggregation strategies out of the box.

5.15. 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>
```

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.jms.CamelJmsSinkConnector
```

The camel-jms sink connector supports 143 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
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

Name	Description	Default	Priority
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
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

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.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.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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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.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

Name	Description	Default	Priority
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
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.transactedInOut	Specifies whether InOut operations (request reply) default to using transacted mode If this flag is set to true, then Spring JmsTemplate will have sessionTransacted set to true, and the acknowledgeMode as transacted on the JmsTemplate used for InOut operations. Note from Spring JMS: that within a JTA transaction, the parameters passed to createQueue, createTopic methods are not taken into account. Depending on the Java EE transaction context, the container makes its own decisions on these values. Analogously, these parameters are not taken into account within a locally managed transaction either, since Spring JMS operates on an existing JMS Session in this case. Setting this flag to true will use a short local JMS transaction when running outside of a managed transaction, and a synchronized local JMS transaction in case of a managed transaction (other than an XA transaction) being present. This has the effect of a local JMS transaction being managed alongside the main transaction (which might be a native JDBC transaction), with the JMS transaction committing right after the main transaction.	false	MEDIUM

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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.transactedInOut	Specifies whether InOut operations (request reply) default to using transacted mode If this flag is set to true, then Spring JmsTemplate will have sessionTransacted set to true, and the acknowledgeMode as transacted on the JmsTemplate used for InOut operations. Note from Spring JMS: that within a JTA transaction, the parameters passed to createQueue, createTopic methods are not taken into account. Depending on the Java EE transaction context, the container makes its own decisions on these values. Analogously, these parameters are not taken into account within a locally managed transaction either, since Spring JMS operates on an existing JMS Session in this case. Setting this flag to true will use a short local JMS transaction when running outside of a managed transaction, and a synchronized local JMS transaction in case of a managed transaction (other than an XA transaction) being present. This has the effect of a local JMS transaction being managed alongside the main transaction (which might be a native JDBC transaction), with the JMS transaction committing right after the main transaction.	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
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

The camel-jms sink connector has no converters out of the box.

The camel-jms sink connector has no transforms out of the box.

The camel-jms sink connector has no aggregation strategies out of the box.

5.16. 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>
```

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.jms.CamelJmsSourceConnector
```

The camel-jms source connector supports 143 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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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.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.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

Name	Description	Default	Priority
camel.source.endpoint.artemisStreamingEnabled	Whether optimizing for Apache Artemis streaming mode.	true	MEDIUM
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

Name	Description	Default	Priority
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
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.waitForProvisionCorrelation ToBeUpdatedCounter	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.waitForProvisionCorrelation ToBeUpdatedThreadSleepingTime	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

Name	Description	Default	Priority
camel.source.endpoint.transactedInOut	Specifies whether InOut operations (request reply) default to using transacted mode If this flag is set to true, then Spring JmsTemplate will have sessionTransacted set to true, and the acknowledgeMode as transacted on the JmsTemplate used for InOut operations. Note from Spring JMS: that within a JTA transaction, the parameters passed to createQueue, createTopic methods are not taken into account. Depending on the Java EE transaction context, the container makes its own decisions on these values. Analogously, these parameters are not taken into account within a locally managed transaction either, since Spring JMS operates on an existing JMS Session in this case. Setting this flag to true will use a short local JMS transaction when running outside of a managed transaction, and a synchronized local JMS transaction in case of a managed transaction (other than an XA transaction) being present. This has the effect of a local JMS transaction being managed alongside the main transaction (which might be a native JDBC transaction), with the JMS transaction committing right after the main transaction.	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
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

Name	Description	Default	Priority
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

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.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

Name	Description	Default	Priority
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.transactedInOut	Specifies whether InOut operations (request reply) default to using transacted mode If this flag is set to true, then Spring JmsTemplate will have sessionTransacted set to true, and the acknowledgeMode as transacted on the JmsTemplate used for InOut operations. Note from Spring JMS: that within a JTA transaction, the parameters passed to createQueue, createTopic methods are not taken into account. Depending on the Java EE transaction context, the container makes its own decisions on these values. Analogously, these parameters are not taken into account within a locally managed transaction either, since Spring JMS operates on an existing JMS Session in this case. Setting this flag to true will use a short local JMS transaction when running outside of a managed transaction, and a synchronized local JMS transaction in case of a managed transaction (other than an XA transaction) being present. This has the effect of a local JMS transaction being managed alongside the main transaction (which might be a native JDBC transaction), with the JMS transaction committing right after the main transaction.	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
camel.component.jms.transactionName	The name of the transaction to use.	null	MEDIUM

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

The camel-jms sink connector has no converters out of the box.

The camel-jms sink connector has no transforms out of the box.

The camel-jms sink connector has no aggregation strategies out of the box.

5.17. CAMEL-MONGODB-KAFKA-CONNECTOR SINK CONFIGURATION

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

To use this Sink connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.mongodb.CamelMongodbSinkConnector
```

The camel-mongodb sink connector supports 26 options, which are listed below.

Name	Description	Default	Priority
camel.sink.path.connectionBean	Sets the connection bean reference used to lookup a client for connecting to a database.	null	HIGH
camel.sink.endpoint.collection	Sets the name of the MongoDB collection to bind to this endpoint	null	MEDIUM
camel.sink.endpoint.collectionIndex	Sets the collection index (JSON FORMAT : { field1 : order1, field2 : order2})	null	MEDIUM
camel.sink.endpoint.createCollection	Create collection during initialisation if it doesn't exist. Default is true.	true	MEDIUM
camel.sink.endpoint.database	Sets the name of the MongoDB database to target	null	MEDIUM
camel.sink.endpoint.mongoConnection	Sets the connection bean used as a client for connecting to a database.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.operation	Sets the operation this endpoint will execute against MongoDB. One of: [findById] [findOneByQuery] [findAll] [findDistinct] [insert] [save] [update] [remove] [bulkWrite] [aggregate] [getDbStats] [getColStats] [count] [command]	null	MEDIUM
camel.sink.endpoint.outputType	Convert the output of the producer to the selected type : DocumentList Document or MongoIterable. DocumentList or MongoIterable applies to findAll and aggregate. Document applies to all other operations. One of: [DocumentList] [Document] [MongoIterable]	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.cursorRegenerationDelay	MongoDB tailable cursors will block until new data arrives. If no new data is inserted, after some time the cursor will be automatically freed and closed by the MongoDB server. The client is expected to regenerate the cursor if needed. This value specifies the time to wait before attempting to fetch a new cursor, and if the attempt fails, how long before the next attempt is made. Default value is 1000ms.	1000L	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.dynamicity	Sets whether this endpoint will attempt to dynamically resolve the target database and collection from the incoming Exchange properties. Can be used to override at runtime the database and collection specified on the otherwise static endpoint URI. It is disabled by default to boost performance. Enabling it will take a minimal performance hit.	false	MEDIUM
camel.sink.endpoint.readPreference	Configure how MongoDB clients route read operations to the members of a replica set. Possible values are PRIMARY, PRIMARY_PREFERRED, SECONDARY, SECONDARY_PREFERRED or NEAREST One of: [PRIMARY] [PRIMARY_PREFERRED] [SECONDARY] [SECONDARY_PREFERRED] [NEAREST]	"PRIMARY"	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.writeConcern	Configure the connection bean with the level of acknowledgment requested from MongoDB for write operations to a standalone mongod, replicaset or cluster. Possible values are ACKNOWLEDGED, W1, W2, W3, UNACKNOWLEDGED, JOURNALED or MAJORITY. One of: [ACKNOWLEDGED] [W1] [W2] [W3] [UNACKNOWLEDGED] [JOURNALED] [MAJORITY]	"ACKNOWLEDGED"	MEDIUM
camel.sink.endpoint.writeResultAsHeader	In write operations, it determines whether instead of returning WriteResult as the body of the OUT message, we transfer the IN message to the OUT and attach the WriteResult as a header.	false	MEDIUM
camel.sink.endpoint.streamFilter	Filter condition for change streams consumer.	null	MEDIUM
camel.sink.endpoint.persistentId	One tail tracking collection can host many trackers for several tailable consumers. To keep them separate, each tracker should have its own unique persistentId.	null	MEDIUM

Name	Description	Default	Priority
camel.sink.endpoint.persistentTailTracking	Enable persistent tail tracking, which is a mechanism to keep track of the last consumed message across system restarts. The next time the system is up, the endpoint will recover the cursor from the point where it last stopped slurping records.	false	MEDIUM
camel.sink.endpoint.tailTrackCollection	Collection where tail tracking information will be persisted. If not specified, MongoDbTailTrackingConfig#DEFAULT_COLLECTION will be used by default.	null	MEDIUM
camel.sink.endpoint.tailTrackDb	Indicates what database the tail tracking mechanism will persist to. If not specified, the current database will be picked by default. Dynamicity will not be taken into account even if enabled, i.e. the tail tracking database will not vary past endpoint initialisation.	null	MEDIUM
camel.sink.endpoint.tailTrackField	Field where the last tracked value will be placed. If not specified, MongoDbTailTrackingConfig#DEFAULT_FIELD will be used by default.	null	MEDIUM
camel.sink.endpoint.tailTrackIncreasingField	Correlation field in the incoming record which is of increasing nature and will be used to position the tailing cursor every time it is generated. The cursor will be (re)created with a query of type: tailTrackIncreasingField greater than lastValue (possibly recovered from persistent tail tracking). Can be of type Integer, Date, String, etc. NOTE: No support for dot notation at the current time, so the field should be at the top level of the document.	null	MEDIUM
camel.component.mongodb.mongoConnection	Shared client used for connection. All endpoints generated from the component will share this connection client.	null	MEDIUM

Name	Description	Default	Priority
camel.component.mongodb.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.mongodb.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM

The camel-mongodb sink connector has no converters out of the box.

The camel-mongodb sink connector has no transforms out of the box.

The camel-mongodb sink connector has no aggregation strategies out of the box.

5.18. CAMEL-MONGODB-KAFKA-CONNECTOR SOURCE CONFIGURATION

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

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.mongodb.CamelMongodbSourceConnector
```

The camel-mongodb source connector supports 29 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.connectionBean	Sets the connection bean reference used to lookup a client for connecting to a database.	null	HIGH

Name	Description	Default	Priority
camel.source.endpoint.collection	Sets the name of the MongoDB collection to bind to this endpoint	null	MEDIUM
camel.source.endpoint.collectionIndex	Sets the collection index (JSON FORMAT : { field1 : order1, field2 : order2})	null	MEDIUM
camel.source.endpoint.createCollection	Create collection during initialisation if it doesn't exist. Default is true.	true	MEDIUM
camel.source.endpoint.database	Sets the name of the MongoDB database to target	null	MEDIUM
camel.source.endpoint.mongoConnection	Sets the connection bean used as a client for connecting to a database.	null	MEDIUM
camel.source.endpoint.operation	Sets the operation this endpoint will execute against MongoDB. One of: [findById] [findOneByQuery] [findAll] [findDistinct] [insert] [save] [update] [remove] [bulkWrite] [aggregate] [getDbStats] [getColStats] [count] [command]	null	MEDIUM
camel.source.endpoint.outputType	Convert the output of the producer to the selected type : DocumentList Document or MongoIterable. DocumentList or MongoIterable applies to findAll and aggregate. Document applies to all other operations. One of: [DocumentList] [Document] [MongoIterable]	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
camel.source.endpoint.consumerType	Consumer type.	null	MEDIUM

Name	Description	Default	Priority
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
camel.source.endpoint.cursorRegenerationDelay	MongoDB tailable cursors will block until new data arrives. If no new data is inserted, after some time the cursor will be automatically freed and closed by the MongoDB server. The client is expected to regenerate the cursor if needed. This value specifies the time to wait before attempting to fetch a new cursor, and if the attempt fails, how long before the next attempt is made. Default value is 1000ms.	1000L	MEDIUM
camel.source.endpoint.dynamicity	Sets whether this endpoint will attempt to dynamically resolve the target database and collection from the incoming Exchange properties. Can be used to override at runtime the database and collection specified on the otherwise static endpoint URI. It is disabled by default to boost performance. Enabling it will take a minimal performance hit.	false	MEDIUM
camel.source.endpoint.readPreference	Configure how MongoDB clients route read operations to the members of a replica set. Possible values are PRIMARY, PRIMARY_PREFERRED, SECONDARY, SECONDARY_PREFERRED or NEAREST One of: [PRIMARY] [PRIMARY_PREFERRED] [SECONDARY] [SECONDARY_PREFERRED] [NEAREST]	"PRIMARY"	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

Name	Description	Default	Priority
camel.source.endpoint.writeConcern	Configure the connection bean with the level of acknowledgment requested from MongoDB for write operations to a standalone mongod, replicaset or cluster. Possible values are ACKNOWLEDGED, W1, W2, W3, UNACKNOWLEDGED, JOURNALED or MAJORITY. One of: [ACKNOWLEDGED] [W1] [W2] [W3] [UNACKNOWLEDGED] [JOURNALED] [MAJORITY]	"ACKNOWLEDGED"	MEDIUM
camel.source.endpoint.writeResultAsHeader	In write operations, it determines whether instead of returning WriteResult as the body of the OUT message, we transfer the IN message to the OUT and attach the WriteResult as a header.	false	MEDIUM
camel.source.endpoint.streamFilter	Filter condition for change streams consumer.	null	MEDIUM
camel.source.endpoint.persistentId	One tail tracking collection can host many trackers for several tailable consumers. To keep them separate, each tracker should have its own unique persistentId.	null	MEDIUM
camel.source.endpoint.persistentTailTracking	Enable persistent tail tracking, which is a mechanism to keep track of the last consumed message across system restarts. The next time the system is up, the endpoint will recover the cursor from the point where it last stopped slurping records.	false	MEDIUM
camel.source.endpoint.tailTrackCollection	Collection where tail tracking information will be persisted. If not specified, MongoDbTailTrackingConfig#DEFAULT_COLLECTION will be used by default.	null	MEDIUM
camel.source.endpoint.tailTrackDb	Indicates what database the tail tracking mechanism will persist to. If not specified, the current database will be picked by default. Dynamicity will not be taken into account even if enabled, i.e. the tail tracking database will not vary past endpoint initialisation.	null	MEDIUM
camel.source.endpoint.tailTrackField	Field where the last tracked value will be placed. If not specified, MongoDbTailTrackingConfig#DEFAULT_FIELD will be used by default.	null	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.tailTrackIncreasingField	Correlation field in the incoming record which is of increasing nature and will be used to position the tailing cursor every time it is generated. The cursor will be (re)created with a query of type: tailTrackIncreasingField greater than lastValue (possibly recovered from persistent tail tracking). Can be of type Integer, Date, String, etc. NOTE: No support for dot notation at the current time, so the field should be at the top level of the document.	null	MEDIUM
camel.component.mongodb.mongoConnection	Shared client used for connection. All endpoints generated from the component will share this connection client.	null	MEDIUM
camel.component.mongodb.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.mongodb.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM

The camel-mongodb sink connector has no converters out of the box.

The camel-mongodb sink connector has no transforms out of the box.

The camel-mongodb sink connector has no aggregation strategies out of the box.

5.19. 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>
```

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.netty.CamelNettySourceConnector
```

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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
camel.source.endpoint.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.source.endpoint.serverExceptionCaughtLogLevel	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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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 .serverClosedChannelE xception 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.component.netty .serverExceptionCaugh tLog 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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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

The camel-netty sink connector has no converters out of the box.

The camel-netty sink connector has no transforms out of the box.

The camel-netty sink connector has no aggregation strategies out of the box.

5.20. 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>
```

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.salesforce.CamelSalesforceSourceConnector
```

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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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. Multiple packages can be separated by comma.	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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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
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

Name	Description	Default	Priority
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

The camel-salesforce sink connector has no converters out of the box.

The camel-salesforce sink connector has no transforms out of the box.

The camel-salesforce sink connector has no aggregation strategies out of the box.

5.21. CAMEL-SLACK-KAFKA-CONNECTOR SOURCE CONFIGURATION

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

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.slack.CamelSlackSourceConnector
```

The camel-slack source connector supports 28 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.channel	The channel name (syntax #name) or slackuser (syntax userName) to send a message directly to an user.	null	HIGH

Name	Description	Default	Priority
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.maxResults	The Max Result for the poll	"10"	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.serverUrl	The Server URL of the Slack instance	"https://slack.com"	MEDIUM
camel.source.endpoint.token	The token to use	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.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

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.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
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

Name	Description	Default	Priority
camel.source.endpoint.scheduler	To use a cron scheduler from either camel-spring or camel-quartz component. Use value spring or quartz for built in scheduler	"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.component.slack.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.slack.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM
camel.component.slack.webhookUrl	The incoming webhook URL	null	MEDIUM

The camel-slack sink connector has no converters out of the box.

The camel-slack sink connector has no transforms out of the box.

The camel-slack sink connector has no aggregation strategies out of the box.

5.22. 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

5.23. CAMEL-TIMER-KAFKA-CONNECTOR SOURCE CONFIGURATION

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

To use this Source connector in Kafka connect you'll need to set the following connector.class

```
connector.class=org.apache.camel.kafkaconnector.timer.CamelTimerSourceConnector
```

The camel-timer source connector supports 17 options, which are listed below.

Name	Description	Default	Priority
camel.source.path.timerName	The name of the timer	null	HIGH

Name	Description	Default	Priority
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.delay	Delay before first event is triggered.	1000L	MEDIUM
camel.source.endpoint.fixedRate	Events take place at approximately regular intervals, separated by the specified period.	false	MEDIUM
camel.source.endpoint.includeMetadata	Whether to include metadata in the exchange such as fired time, timer name, timer count etc. This information is default included.	true	MEDIUM
camel.source.endpoint.period	If greater than 0, generate periodic events every period.	1000L	MEDIUM
camel.source.endpoint.repeatCount	Specifies a maximum limit of number of fires. So if you set it to 1, the timer 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.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
camel.source.endpoint.daemon	Specifies whether or not the thread associated with the timer endpoint runs as a daemon. The default value is true.	true	MEDIUM

Name	Description	Default	Priority
camel.source.endpoint.pattern	Allows you to specify a custom Date pattern to use for setting the time option using URI syntax.	null	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.time	A java.util.Date the first event should be generated. If using the URI, the pattern expected is: yyyy-MM-dd HH:mm:ss or yyyy-MM-dd'T'HH:mm:ss.	null	MEDIUM
camel.source.endpoint.timer	To use a custom Timer	null	MEDIUM
camel.component.timer.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.timer.basicPropertyBinding	Whether the component should use basic property binding (Camel 2.x) or the newer property binding with additional capabilities	false	MEDIUM

The camel-timer sink connector has no converters out of the box.

The camel-timer sink connector has no transforms out of the box.

The camel-timer sink connector has no aggregation strategies out of the box.