Red Hat Integration 2021.Q1

Getting Started with Camel Kafka Connector

TECHNOLOGY PREVIEW only

Last Updated: 2021-11-02

Red Hat Integration 2021.Q1 Getting Started with Camel Kafka Connector

TECHNOLOGY PREVIEW only

Abstract

There are no plans for a release of Camel Kafka Connector beyond the unsupported Technology Preview release. This guide introduces Camel Kafka Connector, describes the Camel Kafka connectors that you can configure, explains how to install into AMQ Streams and Kafka Connect on OpenShift, and how to get started with example Camel Kafka connectors.

Table of Contents

PREFACE

NOTE

There are no plans for a release of Camel Kafka Connector beyond the unsupported Technology Preview release.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

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. There are no plans for a release of Camel Kafka Connector beyond the Technology Preview release. 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.

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 along 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 4.6 or 4.7
- Red Hat Enterprise Linux 8.x
- AMQ Streams 1.6
- Apache Kafka Connect 2.6
- Apache Camel Kafka Connector 0.7.1
- Apache Camel 3.7
- OpenJDK 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
Azure Storage Blob	Sink only
Azure Storage Queue	Sink only
Cassandra Query Language (CQL)	Sink and source

Connector	Sink/source
Elasticsearch	Sink only
File	Sink only
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
RabbitMQ	Sink and source
SQL	Sink and source
SSH	Sink and source
Syslog	Sink and source
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.

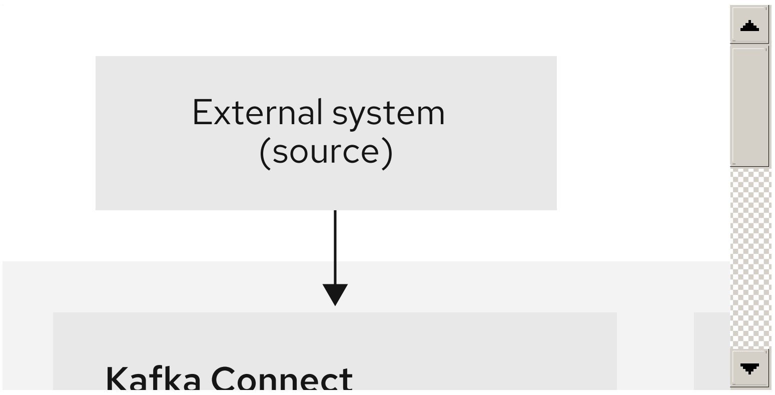
1.3. CAMEL KAFKA CONNECTOR ARCHITECTURE

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 2.6 user documentation

1.4. CAMEL KAFKA CONNECTOR DISTRIBUTIONS

The Camel Kafka Connector distributions are available as part of Red Hat Integration:

Table 1.2. Camel Kafka Connector available distributions

Distribution	Description	Location
Camel Kafka connectors	JAR files for each Camel Kafka connector in **.zip** or **.tar.gz** format	Early-access Maven repository downloads
Maven repository	All Maven artifacts for Camel Kafka Connector	Software Downloads > Red Hat Integration
Source code	All 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.

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.

- Section 2.1, “Authenticating with registry.redhat.io for container images”
- Section 2.2, “Installing AMQ Streams and Kafka Connect S2I on OpenShift”
- Section 2.3, “Deploying Camel Kafka Connector using Kafka Connect S2I on OpenShift”

2.1. AUTHENTICATING WITH REGISTRY.REDHAT.IO FOR CONTAINER IMAGES

Configure authentication with **registry.redhat.io** before you can deploy Camel Kafka Connector container images on OpenShift.

Prerequisites

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

Procedure

1. Log into your OpenShift cluster as administrator:

```
$ 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:

```
$ oc project myproject
```

3. Create a **docker-registry** secret using your Red Hat Customer Portal account, replacing **PULL_SECRET_NAME** with the secret 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 **docker-registry** secret in every OpenShift project namespace that will authenticate to **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 OpenShift pod uses.

5. Link the secret to the **builder** service account to use the secret for pushing and pulling build images:

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

   NOTE

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

Additional resources

For more details on authenticating with Red Hat for container images:

- Red Hat container image 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

- Cluster administrator access to an OpenShift Container Platform cluster.
- Authentication with **registry.redhat.io** using the steps in Section 2.1, "Authenticating with registry.redhat.io for container images".
- 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, 3 Kafka nodes, and **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

- Cluster administrator access to an OpenShift Container Platform cluster.
- AMQ Streams and Kafka Connect with S2I support installed on your OpenShift cluster. For more details, see Section 2.2, "Installing AMQ Streams and Kafka Connect S2I on OpenShift" .
- JAR files for your chosen Camel Kafka connectors downloaded in **.zip** or **tar.gz** format from the Early-access Maven repository. The example in this section uses the AWS2 S3 source connector. The files for each connector must be placed in their own subdirectory, for example:

  ```
  camel-kafka-connector-VERSION/connectors/camel-aws-s3-kafka-connector
  ```

- Access to an Amazon S3 bucket for the example connector in this section.

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.

   NOTE

   In this example, **my-connect-cluster-connect** must match the name of the Kafka Connect S2I cluster that you created in Section 2.2, "Installing AMQ Streams and Kafka Connect S2I on OpenShift".

   ```
   $ oc start-build my-connect-cluster-connect --from-dir=./camel-kafka-connector-
   VERSION/connectors/ --follow
   Uploading directory "camel-kafka-connector-VERSION/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@sha256:12d5ed92510941f1569faa449665e9fc6ea544e67b7ae189ec6b8df434e121f
4
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@sha256:9db57d33df6d0494ea6ee6e4696fcaf79eb81aabeb0bbc180dec5324d33e7ed
a
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","versi
on":"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-connect strimzi.io/use-connector-
resources=true
kafkaconnects2i.kafka.strimzi.io/my-connect-cluster-connect 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 source 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-connect
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.StringAggreg
ator
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.

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
Azure Storage Blob	Camel Azure Storage Blob sink connector	-
Azure Storage Queue	Camel Azure Storage Queue sink 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 SJMS sink connector	Camel SJMS source connector
MongoDB	Camel MongoDB sink connector	Camel MongoDB source connector

Connector	Sink	Source
RabbitMQ	Camel RabbitMQ sink connector	Camel RabbitMQ source connector
SQL	Camel SQL sink connector	Camel SQL source connector
SSH	Camel SSH sink connector	Camel SSH source connector
Syslog	Camel syslog sink connector	Camel syslog source connector
Timer	-	Camel timer source connector

5.1. AMAZON WEB SERVICES KINESIS

5.1.1. camel-aws2-kinesis-kafka-connector sink configuration

Connector Description: Consume and produce records from and to AWS Kinesis Streams using AWS SDK version 2.x.

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 27 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.streamName	Name of the stream	null	true	HIGH
camel.sink.endpoint.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	false	MEDIUM
camel.sink.endpoint.cborEnabled	This option will set the CBOR_ENABLED property during the execution	true	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	false	MEDIUM
camel.sink.endpoint.proxyHost	To define a proxy host when instantiating the Kinesis client	null	false	MEDIUM
camel.sink.endpoint.proxyPort	To define a proxy port when instantiating the Kinesis client	null	false	MEDIUM
camel.sink.endpoint.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	false	MEDIUM
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	false	MEDIUM
camel.sink.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.sink.endpoint.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	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	false	MEDIUM
camel.sink.endpoint.accessKey	Amazon AWS Access Key	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.secretKey	Amazon AWS Secret Key	null	false	MEDIUM
camel.component.aws2-kinesis.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	false	MEDIUM
camel.component.aws2-kinesis.cborEnabled	This option will set the CBOR_ENABLED property during the execution	true	false	MEDIUM
camel.component.aws2-kinesis.configuration	Component configuration	null	false	MEDIUM
camel.component.aws2-kinesis.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	false	MEDIUM
camel.component.aws2-kinesis.proxyHost	To define a proxy host when instantiating the Kinesis client	null	false	MEDIUM
camel.component.aws2-kinesis.proxyPort	To define a proxy port when instantiating the Kinesis client	null	false	MEDIUM
camel.component.aws2-kinesis.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	false	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	false	MEDIUM
camel.component.aws2-kinesis.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.component.aws2-kinesis.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.component.aws2-kinesis.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.aws2-kinesis.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.component.aws2-kinesis.secretKey	Amazon AWS Secret Key	null	false	MEDIUM

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

The camel-aws2-kinesis sink connector supports 0 transforms out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.aws2kinesis.transformers.KinesisRecordDataTransforms
```

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

5.1.2. camel-aws2-kinesis-kafka-connector source configuration

Connector description: Consume and produce records from and to AWS Kinesis Streams using AWS SDK version 2.x.

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 55 options, which are listed below.

Name	Description	Default	Required	Priority
camel.source.path.streamName	Name of the stream	null	true	HIGH
camel.source.endpoint.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	false	MEDIUM
camel.source.endpoint.cborEnabled	This option will set the CBOR_ENABLED property during the execution	true	false	MEDIUM
camel.source.endpoint.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	false	MEDIUM
camel.source.endpoint.proxyHost	To define a proxy host when instantiating the Kinesis client	null	false	MEDIUM
camel.source.endpoint.proxyPort	To define a proxy port when instantiating the Kinesis client	null	false	MEDIUM
camel.source.endpoint.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	false	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	false	MEDIUM
camel.source.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	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	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"	false	MEDIUM
camel.source.endpoint.maxResultsPerRequest	Maximum number of records that will be fetched in each poll	1	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	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	false	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"	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.shardId	Defines which shardId in the Kinesis stream to get records from	null	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	false	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	false	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	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	false	MEDIUM
camel.source.endpoint.backoffIdleThreshold	The number of subsequent idle polls that should happen before the backoffMultipler should kick-in.	null	false	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	false	MEDIUM
camel.source.endpoint.delay	Milliseconds before the next poll.	500L	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.source.endpoint.initialDelay	Milliseconds before the first poll starts.	1000L	false	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	false	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"	false	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	false	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"	false	MEDIUM
camel.source.endpoint.schedulerProperties	To configure additional properties when using a custom scheduler or any of the Quartz, Spring based scheduler.	null	false	MEDIUM
camel.source.endpoint.startScheduler	Whether the scheduler should be auto started.	true	false	MEDIUM
camel.source.endpoint.timeUnit	Time unit for initialDelay and delay options. One of: [NANOSECONDS] [MICROSECONDS] [MILLISECONDS] [SECONDS] [MINUTES] [HOURS] [DAYS]	"MILLISECONDS"	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.useFixedDelay	Controls if fixed delay or fixed rate is used. See ScheduledExecutorService in JDK for details.	true	false	MEDIUM
camel.source.endpoint.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.source.endpoint.secretKey	Amazon AWS Secret Key	null	false	MEDIUM
camel.component.aws2-kinesis.amazonKinesisClient	Amazon Kinesis client to use for all requests for this endpoint	null	false	MEDIUM
camel.component.aws2-kinesis.cborEnabled	This option will set the CBOR_ENABLED property during the execution	true	false	MEDIUM
camel.component.aws2-kinesis.configuration	Component configuration	null	false	MEDIUM
camel.component.aws2-kinesis.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	false	MEDIUM
camel.component.aws2-kinesis.proxyHost	To define a proxy host when instantiating the Kinesis client	null	false	MEDIUM
camel.component.aws2-kinesis.proxyPort	To define a proxy port when instantiating the Kinesis client	null	false	MEDIUM
camel.component.aws2-kinesis.proxyProtocol	To define a proxy protocol when instantiating the Kinesis client One of: [HTTP] [HTTPS]	"HTTPS"	false	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	false	MEDIUM
camel.component.aws2-kinesis.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-kinesis.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	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	false	MEDIUM
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"	false	MEDIUM
camel.component.aws2-kinesis.maxResultsPerRequest	Maximum number of records that will be fetched in each poll	1	false	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	false	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"	false	MEDIUM
camel.component.aws2-kinesis.shardId	Defines which shardId in the Kinesis stream to get records from	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-kinesis.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.aws2-kinesis.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.component.aws2-kinesis.secretKey	Amazon AWS Secret Key	null	false	MEDIUM

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

The camel-aws2-kinesis source connector supports 0 transforms out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.aws2kinesis.transformers.KinesisRecordDataTransforms
```

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

5.2. AMAZON WEB SERVICES S3

5.2.1. camel-aws2-s3-kafka-connector sink configuration

Connector Description: Store and retrieve objects from AWS S3 Storage Service using AWS SDK version 2.x.

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 59 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.bucketNameOrArn	Bucket name or ARN	null	true	HIGH
camel.sink.endpoint.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	false	MEDIUM
camel.sink.endpoint.amazonS3Presigner	An S3 Presigner for Request, used mainly in createDownloadLink operation	null	false	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	false	MEDIUM
camel.sink.endpoint.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	false	MEDIUM
camel.sink.endpoint.pojoRequest	If we want to use a POJO request as body or not	false	false	MEDIUM
camel.sink.endpoint.policy	The policy for this queue to set in the com.amazonaws.services.s3.AmazonS3#setBucketPolicy() method.	null	false	MEDIUM
camel.sink.endpoint.proxyHost	To define a proxy host when instantiating the SQS client	null	false	MEDIUM
camel.sink.endpoint.proxyPort	Specify a proxy port to be used inside the client definition.	null	false	MEDIUM
camel.sink.endpoint.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.sink.endpoint.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	false	MEDIUM
camel.sink.endpoint.useDefaultCredentialsProvider	Set whether the S3 client should expect to load credentials through a default credentials provider or to expect static credentials to be passed in.	false	false	MEDIUM
camel.sink.endpoint.customerAlgorithm	Define the customer algorithm to use in case CustomerKey is enabled	null	false	MEDIUM
camel.sink.endpoint.customerKeyId	Define the id of Customer key to use in case CustomerKey is enabled	null	false	MEDIUM
camel.sink.endpoint.customerKeyMD5	Define the MD5 of Customer key to use in case CustomerKey is enabled	null	false	MEDIUM
camel.sink.endpoint.deleteAfterWrite	Delete file object after the S3 file has been uploaded	false	false	MEDIUM
camel.sink.endpoint.keyName	Setting the key name for an element in the bucket through endpoint parameter	null	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	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	false	MEDIUM
camel.sink.endpoint.partSize	Setup the partSize which is used in multi part upload, the default size is 25M.	26214400L	false	MEDIUM
camel.sink.endpoint.storageClass	The storage class to set in the com.amazonaws.services.s3.model.PutObjectRequest request.	null	false	MEDIUM
camel.sink.endpoint.awsKMSKeyId	Define the id of KMS key to use in case KMS is enabled	null	false	MEDIUM
camel.sink.endpoint.useAwsKMS	Define if KMS must be used or not	false	false	MEDIUM
camel.sink.endpoint.useCustomerKey	Define if Customer Key must be used or not	false	false	MEDIUM
camel.sink.endpoint.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.sink.endpoint.secretKey	Amazon AWS Secret Key	null	false	MEDIUM
camel.component.aws2-s3.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	false	MEDIUM
camel.component.aws2-s3.amazonS3Presigner	An S3 Presigner for Request, used mainly in createDownloadLink operation	null	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.component.aws2-s3.configuration	The component configuration	null	false	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	false	MEDIUM
camel.component.aws2-s3.pojoRequest	If we want to use a POJO request as body or not	false	false	MEDIUM
camel.component.aws2-s3.policy	The policy for this queue to set in the com.amazonaws.services.s3.Amazon S3#setBucketPolicy() method.	null	false	MEDIUM
camel.component.aws2-s3.proxyHost	To define a proxy host when instantiating the SQS client	null	false	MEDIUM
camel.component.aws2-s3.proxyPort	Specify a proxy port to be used inside the client definition.	null	false	MEDIUM
camel.component.aws2-s3.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	false	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	false	MEDIUM
camel.component.aws2-s3.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.component.aws2-s3.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-s3.useDefaultCredentials Provider	Set whether the S3 client should expect to load credentials through a default credentials provider or to expect static credentials to be passed in.	false	false	MEDIUM
camel.component.aws2-s3.customerAlgorithm	Define the customer algorithm to use in case CustomerKey is enabled	null	false	MEDIUM
camel.component.aws2-s3.customerKeyId	Define the id of Customer key to use in case CustomerKey is enabled	null	false	MEDIUM
camel.component.aws2-s3.customerKeyMD5	Define the MD5 of Customer key to use in case CustomerKey is enabled	null	false	MEDIUM
camel.component.aws2-s3.deleteAfterWrite	Delete file object after the S3 file has been uploaded	false	false	MEDIUM
camel.component.aws2-s3.keyName	Setting the key name for an element in the bucket through endpoint parameter	null	false	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	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	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.component.aws2-s3.partSize	Setup the partSize which is used in multi part upload, the default size is 25M.	26214400L	false	MEDIUM
camel.component.aws2-s3.storageClass	The storage class to set in the com.amazonaws.services.s3.model.PutObjectRequest request.	null	false	MEDIUM
camel.component.aws2-s3.awsKMSKeyId	Define the id of KMS key to use in case KMS is enabled	null	false	MEDIUM
camel.component.aws2-s3.useAwsKMS	Define if KMS must be used or not	false	false	MEDIUM
camel.component.aws2-s3.useCustomerKey	Define if Customer Key must be used or not	false	false	MEDIUM
camel.component.aws2-s3.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.aws2-s3.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.component.aws2-s3.secretKey	Amazon AWS Secret Key	null	false	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 supports 1 transforms out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.aws2s3.transformers.S3ObjectTransforms
```

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

```
org.apache.camel.kafkaconnector.aws2s3.aggregation.NewlineAggregationStrategy
```

5.2.2. camel-aws2-s3-kafka-connector source configuration

Connector description: Store and retrieve objects from AWS S3 Storage Service using AWS SDK version 2.x.

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	Required	Priority
camel.source.path.bucketNameOrArn	Bucket name or ARN	null	true	HIGH
camel.source.endpoint.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	false	MEDIUM
camel.source.endpoint.amazonS3Presigner	An S3 Presigner for Request, used mainly in createDownloadLink operation	null	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.overrideEndpoint	Set the need for overidding the endpoint. This option needs to be used in combination with uriEndpointOverride option	false	false	MEDIUM
camel.source.endpoint.pojoRequest	If we want to use a POJO request as body or not	false	false	MEDIUM
camel.source.endpoint.policy	The policy for this queue to set in the com.amazonaws.services.s3.AmazonS3#setBucketPolicy() method.	null	false	MEDIUM
camel.source.endpoint.proxyHost	To define a proxy host when instantiating the SQS client	null	false	MEDIUM
camel.source.endpoint.proxyPort	Specify a proxy port to be used inside the client definition.	null	false	MEDIUM
camel.source.endpoint.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	false	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	false	MEDIUM
camel.source.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.source.endpoint.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	false	MEDIUM
camel.source.endpoint.useDefaultCredentialsProvider	Set whether the S3 client should expect to load credentials through a default credentials provider or to expect static credentials to be passed in.	false	false	MEDIUM
camel.source.endpoint.customerAlgorithm	Define the customer algorithm to use in case CustomerKey is enabled	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.customerKeyId	Define the id of Customer key to use in case CustomerKey is enabled	null	false	MEDIUM
camel.source.endpoint.customerKeyMD5	Define the MD5 of Customer key to use in case CustomerKey is enabled	null	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	false	MEDIUM
camel.source.endpoint.deleteAfterRead	Delete objects from S3 after they have been retrieved. The delete is only performed if the Exchange is committed. If a rollback occurs, the object is not deleted. If this option is false, then the same objects will be retrieve over and over again on the polls. Therefore you need to use the Idempotent Consumer EIP in the route to filter out duplicates. You can filter using the AWS2S3Constants#BUCKET_NAME and AWS2S3Constants#KEY headers, or only the AWS2S3Constants#KEY header.	true	false	MEDIUM
camel.source.endpoint.delimiter	The delimiter which is used in the com.amazonaws.services.s3.model.ListObjectsRequest to only consume objects we are interested in.	null	false	MEDIUM
camel.source.endpoint.destinationBucket	Define the destination bucket where an object must be moved when moveAfterRead is set to true.	null	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.destinationBucketSuffix	Define the destination bucket suffix to use when an object must be moved and moveAfterRead is set to true.	null	false	MEDIUM
camel.source.endpoint.doneFileName	If provided, Camel will only consume files if a done file exists.	null	false	MEDIUM
camel.source.endpoint.fileName	To get the object from the bucket with the given file name	null	false	MEDIUM
camel.source.endpoint.includeBody	If it is true, the S3Object exchange will be consumed and put into the body and closed. If false the S3Object stream will be put raw into the body and the headers will be set with the S3 object metadata. This option is strongly related to autocloseBody option. In case of setting includeBody to true because the S3Object stream will be consumed then it will also be closed in case of includeBody false then it will be up to the caller to close the S3Object stream. However setting autocloseBody to true when includeBody is false it will schedule to close the S3Object stream automatically on exchange completion.	true	false	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	false	MEDIUM
camel.source.endpoint.maxConnections	Set the maxConnections parameter in the S3 client configuration	60	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.source.endpoint.prefix	The prefix which is used in the com.amazonaws.services.s3.model.ListObjectsRequest to only consume objects we are interested in.	null	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	false	MEDIUM
camel.source.endpoint.autocloseBody	If this option is true and includeBody is false, 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 false 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	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	false	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	false	MEDIUM

Name	Description	Default	Required	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	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	false	MEDIUM
camel.source.endpoint.backoffIdleThreshold	The number of subsequent idle polls that should happen before the backoffMultipler should kick-in.	null	false	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	false	MEDIUM
camel.source.endpoint.delay	Milliseconds before the next poll.	500L	false	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	false	MEDIUM
camel.source.endpoint.initialDelay	Milliseconds before the first poll starts.	1000L	false	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	false	MEDIUM

Name	Description	Default	Required	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"	false	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	false	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"	false	MEDIUM
camel.source.endpoint.schedulerProperties	To configure additional properties when using a custom scheduler or any of the Quartz, Spring based scheduler.	null	false	MEDIUM
camel.source.endpoint.startScheduler	Whether the scheduler should be auto started.	true	false	MEDIUM
camel.source.endpoint.timeUnit	Time unit for initialDelay and delay options. One of: [NANOSECONDS] [MICROSECONDS] [MILLISECONDS] [SECONDS] [MINUTES] [HOURS] [DAYS]	"MILLISECONDS"	false	MEDIUM
camel.source.endpoint.useFixedDelay	Controls if fixed delay or fixed rate is used. See ScheduledExecutorService in JDK for details.	true	false	MEDIUM
camel.source.endpoint.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.source.endpoint.secretKey	Amazon AWS Secret Key	null	false	MEDIUM
camel.component.aws2-s3.amazonS3Client	Reference to a com.amazonaws.services.s3.AmazonS3 in the registry.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-s3.amazonS3Presigner	An S3 Presigner for Request, used mainly in createDownloadLink operation	null	false	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	false	MEDIUM
camel.component.aws2-s3.configuration	The component configuration	null	false	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	false	MEDIUM
camel.component.aws2-s3.pojoRequest	If we want to use a POJO request as body or not	false	false	MEDIUM
camel.component.aws2-s3.policy	The policy for this queue to set in the com.amazonaws.services.s3.AmazonS3#setBucketPolicy() method.	null	false	MEDIUM
camel.component.aws2-s3.proxyHost	To define a proxy host when instantiating the SQS client	null	false	MEDIUM
camel.component.aws2-s3.proxyPort	Specify a proxy port to be used inside the client definition.	null	false	MEDIUM
camel.component.aws2-s3.proxyProtocol	To define a proxy protocol when instantiating the S3 client One of: [HTTP] [HTTPS]	"HTTPS"	false	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	false	MEDIUM
camel.component.aws2-s3.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-s3.uriEndpointOverride	Set the overriding uri endpoint. This option needs to be used in combination with overrideEndpoint option	null	false	MEDIUM
camel.component.aws2-s3.useDefaultCredentials Provider	Set whether the S3 client should expect to load credentials through a default credentials provider or to expect static credentials to be passed in.	false	false	MEDIUM
camel.component.aws2-s3.customerAlgorithm	Define the customer algorithm to use in case CustomerKey is enabled	null	false	MEDIUM
camel.component.aws2-s3.customerKeyId	Define the id of Customer key to use in case CustomerKey is enabled	null	false	MEDIUM
camel.component.aws2-s3.customerKeyMD5	Define the MD5 of Customer key to use in case CustomerKey is enabled	null	false	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	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-s3.delimiter	The delimiter which is used in the com.amazonaws.services.s3.model.ListObjectsRequest to only consume objects we are interested in.	null	false	MEDIUM
camel.component.aws2-s3.destinationBucket	Define the destination bucket where an object must be moved when moveAfterRead is set to true.	null	false	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	false	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	false	MEDIUM
camel.component.aws2-s3.doneFileName	If provided, Camel will only consume files if a done file exists.	null	false	MEDIUM
camel.component.aws2-s3.fileName	To get the object from the bucket with the given file name	null	false	MEDIUM
camel.component.aws2-s3.includeBody	If it is true, the S3Object exchange will be consumed and put into the body and closed. If false the S3Object stream will be put raw into the body and the headers will be set with the S3 object metadata. This option is strongly related to autocloseBody option. In case of setting includeBody to true because the S3Object stream will be consumed then it will also be closed in case of includeBody false then it will be up to the caller to close the S3Object stream. However setting autocloseBody to true when includeBody is false it will schedule to close the S3Object stream automatically on exchange completion.	true	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.component.aws2-s3.prefix	The prefix which is used in the com.amazonaws.services.s3.model.ListObjectsRequest to only consume objects we are interested in.	null	false	MEDIUM
camel.component.aws2-s3.autocloseBody	If this option is true and includeBody is false, 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 false 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	false	MEDIUM
camel.component.aws2-s3.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.aws2-s3.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.component.aws2-s3.secretKey	Amazon AWS Secret Key	null	false	MEDIUM

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

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

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

```
org.apache.camel.kafkaconnector.aws2s3.transformers.S3ObjectTransforms
```

The camel-aws2-s3 source connector supports 1 aggregation strategies out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.aws2s3.aggregation.NewlineAggregationStrategy
```

5.3. AMAZON WEB SERVICES SNS

5.3.1. camel-aws2-sns-kafka-connector sink configuration

Connector Description: Send messages to an AWS Simple Notification Topic using AWS SDK version 2.x.

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 44 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.topicNameOrArn	Topic name or ARN	null	true	HIGH
camel.sink.endpoint.amazonSNSClient	To use the AmazonSNS as the client	null	false	MEDIUM
camel.sink.endpoint.autoCreateTopic	Setting the autocreation of the topic	true	false	MEDIUM
camel.sink.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to map headers to/from Camel.	null	false	MEDIUM
camel.sink.endpoint.kmsMasterKeyId	The ID of an AWS-managed customer master key (CMK) for Amazon SNS or a custom CMK.	null	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.sink.endpoint.messageDeduplicationIdStrategy	Only for FIFO Topic. 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"	false	MEDIUM
camel.sink.endpoint.messageGroupIdStrategy	Only for FIFO Topic. 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	false	MEDIUM
camel.sink.endpoint.messageStructure	The message structure to use such as json	null	false	MEDIUM
camel.sink.endpoint.policy	The policy for this topic. Is loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	false	MEDIUM
camel.sink.endpoint.proxyHost	To define a proxy host when instantiating the SNS client	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.proxyPort	To define a proxy port when instantiating the SNS client	null	false	MEDIUM
camel.sink.endpoint.proxyProtocol	To define a proxy protocol when instantiating the SNS client One of: [HTTP] [HTTPS]	"HTTPS"	false	MEDIUM
camel.sink.endpoint.queueUrl	The queueUrl to subscribe to	null	false	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	false	MEDIUM
camel.sink.endpoint.serverSideEncryptionEnabled	Define if Server Side Encryption is enabled or not on the topic	false	false	MEDIUM
camel.sink.endpoint.subject	The subject which is used if the message header 'CamelAwsSnsSubject' is not present.	null	false	MEDIUM
camel.sink.endpoint.subscribeSNStoSQS	Define if the subscription between SNS Topic and SQS must be done or not	false	false	MEDIUM
camel.sink.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.sink.endpoint.useDefaultCredentialsProvider	Set whether the SNS client should expect to load credentials on an AWS infra instance or to expect static credentials to be passed in.	false	false	MEDIUM
camel.sink.endpoint.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.sink.endpoint.secretKey	Amazon AWS Secret Key	null	false	MEDIUM
camel.component.aws2-sns.amazonSNSClient	To use the AmazonSNS as the client	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-sns.autoCreateTopic	Setting the autocreation of the topic	true	false	MEDIUM
camel.component.aws2-sns.configuration	Component configuration	null	false	MEDIUM
camel.component.aws2-sns.kmsMasterKeyId	The ID of an AWS-managed customer master key (CMK) for Amazon SNS or a custom CMK.	null	false	MEDIUM
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	false	MEDIUM
camel.component.aws2-sns.messageDeduplicationId Strategy	Only for FIFO Topic. 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"	false	MEDIUM
camel.component.aws2-sns.messageGroupIdStrategy	Only for FIFO Topic. 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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-sns.messageStructure	The message structure to use such as json	null	false	MEDIUM
camel.component.aws2-sns.policy	The policy for this topic. Is loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	false	MEDIUM
camel.component.aws2-sns.proxyHost	To define a proxy host when instantiating the SNS client	null	false	MEDIUM
camel.component.aws2-sns.proxyPort	To define a proxy port when instantiating the SNS client	null	false	MEDIUM
camel.component.aws2-sns.proxyProtocol	To define a proxy protocol when instantiating the SNS client One of: [HTTP] [HTTPS]	"HTTPS"	false	MEDIUM
camel.component.aws2-sns.queueUrl	The queueUrl to subscribe to	null	false	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	false	MEDIUM
camel.component.aws2-sns.serverSideEncryptio n Enabled	Define if Server Side Encryption is enabled or not on the topic	false	false	MEDIUM
camel.component.aws2-sns.subject	The subject which is used if the message header 'CamelAwsSnsSubject' is not present.	null	false	MEDIUM
camel.component.aws2-sns.subscribeSNStoSQS	Define if the subscription between SNS Topic and SQS must be done or not	false	false	MEDIUM
camel.component.aws2-sns.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-sns.useDefaultCredentials Provider	Set whether the SNS client should expect to load credentials on an AWS infra instance or to expect static credentials to be passed in.	false	false	MEDIUM
camel.component.aws2-sns.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.aws2-sns.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.component.aws2-sns.secretKey	Amazon AWS Secret Key	null	false	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.

5.4. AMAZON WEB SERVICES SQS

5.4.1. camel-aws2-sqs-kafka-connector sink configuration

Connector Description: Sending and receive messages to/from AWS SQS service using AWS SDK version 2.x.

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 54 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.queueNameOrArn	Queue name or ARN	null	true	HIGH
camel.sink.endpoint.amazonAWSHost	The hostname of the Amazon AWS cloud.	"amazonaws.com"	false	MEDIUM
camel.sink.endpoint.amazonSQSClient	To use the AmazonSQS as client	null	false	MEDIUM
camel.sink.endpoint.autoCreateQueue	Setting the autocreation of the queue	true	false	MEDIUM
camel.sink.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to map headers to/from Camel.	null	false	MEDIUM
camel.sink.endpoint.protocol	The underlying protocol used to communicate with SQS	"https"	false	MEDIUM
camel.sink.endpoint.proxyProtocol	To define a proxy protocol when instantiating the SQS client One of: [HTTP] [HTTPS]	"HTTPS"	false	MEDIUM
camel.sink.endpoint.queueOwnerAWSAccountId	Specify the queue owner aws account id when you need to connect the queue with different account owner.	null	false	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	false	MEDIUM
camel.sink.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.sink.endpoint.useDefaultCredentialsProvider	Set whether the SQS client should expect to load credentials on an AWS infra instance or to expect static credentials to be passed in.	false	false	MEDIUM
camel.sink.endpoint.delaySeconds	Delay sending messages for a number of seconds.	null	false	MEDIUM

Name	Description	Default	Required	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	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"	false	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	false	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	false	MEDIUM
camel.sink.endpoint.delayQueue	Define if you want to apply delaySeconds option to the queue or on single messages	false	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.sink.endpoint.proxyHost	To define a proxy host when instantiating the SQS client	null	false	MEDIUM
camel.sink.endpoint.proxyPort	To define a proxy port when instantiating the SQS client	null	false	MEDIUM
camel.sink.endpoint.maximumMessageSize	The maximumMessageSize (in bytes) an SQS message can contain for this queue.	null	false	MEDIUM
camel.sink.endpoint.messageRetentionPeriod	The messageRetentionPeriod (in seconds) a message will be retained by SQS for this queue.	null	false	MEDIUM
camel.sink.endpoint.policy	The policy for this queue. It can be loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	false	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	false	MEDIUM
camel.sink.endpoint.redrivePolicy	Specify the policy that send message to DeadLetter queue. See detail at Amazon docs.	null	false	MEDIUM
camel.sink.endpoint.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.sink.endpoint.secretKey	Amazon AWS Secret Key	null	false	MEDIUM
camel.component.aws2-sqs.amazonAWSHost	The hostname of the Amazon AWS cloud.	"amazonaws.com"	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-sqs.amazonSQSClient	To use the AmazonSQS as client	null	false	MEDIUM
camel.component.aws2-sqs.autoCreateQueue	Setting the autocreation of the queue	true	false	MEDIUM
camel.component.aws2-sqs.configuration	The AWS SQS default configuration	null	false	MEDIUM
camel.component.aws2-sqs.protocol	The underlying protocol used to communicate with SQS	"https"	false	MEDIUM
camel.component.aws2-sqs.proxyProtocol	To define a proxy protocol when instantiating the SQS client One of: [HTTP] [HTTPS]	"HTTPS"	false	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	false	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	false	MEDIUM
camel.component.aws2-sqs.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.component.aws2-sqs.useDefaultCredentials Provider	Set whether the SQS client should expect to load credentials on an AWS infra instance or to expect static credentials to be passed in.	false	false	MEDIUM
camel.component.aws2-sqs.delaySeconds	Delay sending messages for a number of seconds.	null	false	MEDIUM

Name	Description	Default	Required	Priority
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	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"	false	MEDIUM
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	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-sqs.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.aws2-sqs.delayQueue	Define if you want to apply delaySeconds option to the queue or on single messages	false	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	false	MEDIUM
camel.component.aws2-sqs.proxyHost	To define a proxy host when instantiating the SQS client	null	false	MEDIUM
camel.component.aws2-sqs.proxyPort	To define a proxy port when instantiating the SQS client	null	false	MEDIUM
camel.component.aws2-sqs.maximumMessageSize	The maximumMessageSize (in bytes) an SQS message can contain for this queue.	null	false	MEDIUM
camel.component.aws2-sqs.messageRetentionPeriod	The messageRetentionPeriod (in seconds) a message will be retained by SQS for this queue.	null	false	MEDIUM
camel.component.aws2-sqs.policy	The policy for this queue. It can be loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	false	MEDIUM
camel.component.aws2-sqs.receiveMessageWait Time Seconds	If you do not specify WaitTimeSeconds in the request, the queue attribute ReceiveMessageWaitTimeSeconds is used to determine how long to wait.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-sqs.redrivePolicy	Specify the policy that send message to DeadLetter queue. See detail at Amazon docs.	null	false	MEDIUM
camel.component.aws2-sqs.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.component.aws2-sqs.secretKey	Amazon AWS Secret Key	null	false	MEDIUM

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

The camel-aws2-sqs sink connector supports 0 transforms out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.aws2sqs.transformers.SQSKeySetterTransforms
```

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

5.4.2. camel-aws2-sqs-kafka-connector source configuration

Connector description: Sending and receive messages to/from AWS SQS service using AWS SDK version 2.x.

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 89 options, which are listed below.

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

Name	Description	Default	Required	Priority
camel.source.endpoint.amazonSQSClient	To use the AmazonSQS as client	null	false	MEDIUM
camel.source.endpoint.autoCreateQueue	Setting the autocreation of the queue	true	false	MEDIUM
camel.source.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to map headers to/from Camel.	null	false	MEDIUM
camel.source.endpoint.protocol	The underlying protocol used to communicate with SQS	"https"	false	MEDIUM
camel.source.endpoint.proxyProtocol	To define a proxy protocol when instantiating the SQS client One of: [HTTP] [HTTPS]	"HTTPS"	false	MEDIUM
camel.source.endpoint.queueOwnerAWSAccountId	Specify the queue owner aws account id when you need to connect the queue with different account owner.	null	false	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	false	MEDIUM
camel.source.endpoint.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.source.endpoint.useDefaultCredentialsProvider	Set whether the SQS client should expect to load credentials on an AWS infra instance or to expect static credentials to be passed in.	false	false	MEDIUM
camel.source.endpoint.attributeNames	A list of attribute names to receive when consuming. Multiple names can be separated by comma.	null	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.source.endpoint.concurrentConsumers	Allows you to use multiple threads to poll the sqs queue to increase throughput	1	false	MEDIUM
camel.source.endpoint.defaultVisibilityTimeout	The default visibility timeout (in seconds)	null	false	MEDIUM
camel.source.endpoint.deleteAfterRead	Delete message from SQS after it has been read	true	false	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	false	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	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.kmsMasterKeyId	The ID of an AWS-managed customer master key (CMK) for Amazon SQS or a custom CMK.	null	false	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	false	MEDIUM
camel.source.endpoint.messageAttributeNames	A list of message attribute names to receive when consuming. Multiple names can be separated by comma.	null	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	false	MEDIUM
camel.source.endpoint.serverSideEncryptionEnabled	Define if Server Side Encryption is enabled or not on the queue	false	false	MEDIUM
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	false	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	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	false	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	false	MEDIUM
camel.source.endpoint.delayQueue	Define if you want to apply delaySeconds option to the queue or on single messages	false	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	false	MEDIUM
camel.source.endpoint.proxyHost	To define a proxy host when instantiating the SQS client	null	false	MEDIUM
camel.source.endpoint.proxyPort	To define a proxy port when instantiating the SQS client	null	false	MEDIUM
camel.source.endpoint.maximumMessageSize	The maximumMessageSize (in bytes) an SQS message can contain for this queue.	null	false	MEDIUM
camel.source.endpoint.messageRetentionPeriod	The messageRetentionPeriod (in seconds) a message will be retained by SQS for this queue.	null	false	MEDIUM
camel.source.endpoint.policy	The policy for this queue. It can be loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.source.endpoint.redrivePolicy	Specify the policy that send message to DeadLetter queue. See detail at Amazon docs.	null	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	false	MEDIUM
camel.source.endpoint.backoffIdleThreshold	The number of subsequent idle polls that should happen before the backoffMultipler should kick-in.	null	false	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	false	MEDIUM
camel.source.endpoint.delay	Milliseconds before the next poll.	500L	false	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	false	MEDIUM
camel.source.endpoint.initialDelay	Milliseconds before the first poll starts.	1000L	false	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	false	MEDIUM

Name	Description	Default	Required	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"	false	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	false	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"	false	MEDIUM
camel.source.endpoint.schedulerProperties	To configure additional properties when using a custom scheduler or any of the Quartz, Spring based scheduler.	null	false	MEDIUM
camel.source.endpoint.startScheduler	Whether the scheduler should be auto started.	true	false	MEDIUM
camel.source.endpoint.timeUnit	Time unit for initialDelay and delay options. One of: [NANOSECONDS] [MICROSECONDS] [MILLISECONDS] [SECONDS] [MINUTES] [HOURS] [DAYS]	"MILLISECONDS"	false	MEDIUM
camel.source.endpoint.useFixedDelay	Controls if fixed delay or fixed rate is used. See ScheduledExecutorService in JDK for details.	true	false	MEDIUM
camel.source.endpoint.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.source.endpoint.secretKey	Amazon AWS Secret Key	null	false	MEDIUM
camel.component.aws2-sqs.amazonAWSHost	The hostname of the Amazon AWS cloud.	"amazonaws.com"	false	MEDIUM
camel.component.aws2-sqs.amazonSQSClient	To use the AmazonSQS as client	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-sqs.autoCreateQueue	Setting the autocreation of the queue	true	false	MEDIUM
camel.component.aws2-sqs.configuration	The AWS SQS default configuration	null	false	MEDIUM
camel.component.aws2-sqs.protocol	The underlying protocol used to communicate with SQS	"https"	false	MEDIUM
camel.component.aws2-sqs.proxyProtocol	To define a proxy protocol when instantiating the SQS client One of: [HTTP] [HTTPS]	"HTTPS"	false	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	false	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	false	MEDIUM
camel.component.aws2-sqs.trustAllCertificates	If we want to trust all certificates in case of overriding the endpoint	false	false	MEDIUM
camel.component.aws2-sqs.useDefaultCredentials Provider	Set whether the SQS client should expect to load credentials on an AWS infra instance or to expect static credentials to be passed in.	false	false	MEDIUM
camel.component.aws2-sqs.attributeNames	A list of attribute names to receive when consuming. Multiple names can be separated by comma.	null	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.component.aws2-sqs.concurrentConsumers	Allows you to use multiple threads to poll the sqs queue to increase throughput	1	false	MEDIUM
camel.component.aws2-sqs.defaultVisibilityTimeout	The default visibility timeout (in seconds)	null	false	MEDIUM
camel.component.aws2-sqs.deleteAfterRead	Delete message from SQS after it has been read	true	false	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	false	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	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.aws2-sqs.kmsMasterKeyId	The ID of an AWS-managed customer master key (CMK) for Amazon SQS or a custom CMK.	null	false	MEDIUM
camel.component.aws2-sqs.messageAttributeNames	A list of message attribute names to receive when consuming. Multiple names can be separated by comma.	null	false	MEDIUM
camel.component.aws2-sqs.serverSideEncryption Enabled	Define if Server Side Encryption is enabled or not on the queue	false	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. SetQueueAttributesRequest. This only make sense if its different from defaultVisibilityTimeout. It changes the queue visibility timeout attribute permanently.	null	false	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	false	MEDIUM
camel.component.aws2-sqs.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.aws2-sqs.delayQueue	Define if you want to apply delaySeconds option to the queue or on single messages	false	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.component.aws2-sqs.proxyHost	To define a proxy host when instantiating the SQS client	null	false	MEDIUM
camel.component.aws2-sqs.proxyPort	To define a proxy port when instantiating the SQS client	null	false	MEDIUM
camel.component.aws2-sqs.maximumMessageSize	The maximumMessageSize (in bytes) an SQS message can contain for this queue.	null	false	MEDIUM
camel.component.aws2-sqs.messageRetentionPeriod	The messageRetentionPeriod (in seconds) a message will be retained by SQS for this queue.	null	false	MEDIUM
camel.component.aws2-sqs.policy	The policy for this queue. It can be loaded by default from classpath, but you can prefix with classpath:, file:, or http: to load the resource from different systems.	null	false	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	false	MEDIUM
camel.component.aws2-sqs.redrivePolicy	Specify the policy that send message to DeadLetter queue. See detail at Amazon docs.	null	false	MEDIUM
camel.component.aws2-sqs.accessKey	Amazon AWS Access Key	null	false	MEDIUM
camel.component.aws2-sqs.secretKey	Amazon AWS Secret Key	null	false	MEDIUM

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

The camel-aws2-sqs source connector supports 0 transforms out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.aws2sqs.transformers.SQSKeySetterTransforms
```

–

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

5.5. AZURE

5.5.1. camel-azure-storage-blob-kafka-connector sink configuration

Connector Description: Store and retrieve blobs from Azure Storage Blob Service using SDK v12.

When using camel-azure-storage-blob-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-azure-storage-blob-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.azurestorageblob.CamelAzurestorageblobSinkConn
ector
```

The camel-azure-storage-blob sink connector supports 55 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.accountName	Azure account name to be used for authentication with azure blob services	null	false	MEDIUM
camel.sink.path.containerName	The blob container name	null	false	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	false	MEDIUM
camel.sink.endpoint.blobName	The blob name, to consume specific blob from a container. However on producer, is only required for the operations on the blob level	null	false	MEDIUM
camel.sink.endpoint.blobOffset	Set the blob offset for the upload or download operations, default is 0	0L	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.blobServiceClient	Client to a storage account. This client does not hold any state about a particular storage account but is instead a convenient way of sending off appropriate requests to the resource on the service. It may also be used to construct URLs to blobs and containers. This client contains operations on a service account. Operations on a container are available on BlobContainerClient through getBlobContainerClient(String), and operations on a blob are available on BlobClient through getBlobContainerClient(String).getBlobClient(String).	null	false	MEDIUM
camel.sink.endpoint.blobType	The blob type in order to initiate the appropriate settings for each blob type One of: [blockblob] [appendblob] [pageblob]	"blockblob"	false	MEDIUM
camel.sink.endpoint.closeStreamAfterRead	Close the stream after read or keep it open, default is true	true	false	MEDIUM
camel.sink.endpoint.credentials	StorageSharedKeyCredential can be injected to create the azure client, this holds the important authentication information	null	false	MEDIUM
camel.sink.endpoint.dataCount	How many bytes to include in the range. Must be greater than or equal to 0 if specified.	null	false	MEDIUM
camel.sink.endpoint.fileDir	The file directory where the downloaded blobs will be saved to, this can be used in both, producer and consumer	null	false	MEDIUM
camel.sink.endpoint.maxResultsPerPage	Specifies the maximum number of blobs to return, including all BlobPrefix elements. If the request does not specify maxResultsPerPage or specifies a value greater than 5,000, the server will return up to 5,000 items.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.maxRetryRequests	Specifies the maximum number of additional HTTP Get requests that will be made while reading the data from a response body.	0	false	MEDIUM
camel.sink.endpoint.prefix	Filters the results to return only blobs whose names begin with the specified prefix. May be null to return all blobs.	null	false	MEDIUM
camel.sink.endpoint.regex	Filters the results to return only blobs whose names match the specified regular expression. May be null to return all if both prefix and regex are set, regex takes the priority and prefix is ignored.	null	false	MEDIUM
camel.sink.endpoint.serviceClient	Client to a storage account. This client does not hold any state about a particular storage account but is instead a convenient way of sending off appropriate requests to the resource on the service. It may also be used to construct URLs to blobs and containers. This client contains operations on a service account. Operations on a container are available on BlobContainerClient through BlobServiceClient#getBlobContainerClient(String), and operations on a blob are available on BlobClient through BlobContainerClient#getBlobClient(String).	null	false	MEDIUM
camel.sink.endpoint.timeout	An optional timeout value beyond which a RuntimeException will be raised.	null	false	MEDIUM
camel.sink.endpoint.blobSequenceNumber	A user-controlled value that you can use to track requests. The value of the sequence number must be between 0 and 263 - 1.The default value is 0.	"0"	false	MEDIUM
camel.sink.endpoint.blockListType	Specifies which type of blocks to return. One of: [committed] [uncommitted] [all]	"COMMITTED"	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.closeStreamAfterWrite	Close the stream after write or keep it open, default is true	true	false	MEDIUM
camel.sink.endpoint.commitBlockListLater	When is set to true, the staged blocks will not be committed directly.	true	false	MEDIUM
camel.sink.endpoint.createAppendBlob	When is set to true, the append blocks will be created when committing append blocks.	true	false	MEDIUM
camel.sink.endpoint.createPageBlob	When is set to true, the page blob will be created when uploading page blob.	true	false	MEDIUM
camel.sink.endpoint.downloadLinkExpiration	Override the default expiration (millis) of URL download link.	null	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	false	MEDIUM
camel.sink.endpoint.operation	The blob operation that can be used with this component on the producer One of: [listBlobContainers] [createBlobContainer] [deleteBlobContainer] [listBlobs] [getBlob] [deleteBlob] [downloadBlobToFile] [downloadLink] [uploadBlockBlob] [stageBlockBlobList] [commitBlobBlockList] [getBlobBlockList] [createAppendBlob] [commitAppendBlob] [createPageBlob] [uploadPageBlob] [resizePageBlob] [clearPageBlob] [getPageBlobRanges]	"listBlobContainers"	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.pageBlobSize	Specifies the maximum size for the page blob, up to 8 TB. The page blob size must be aligned to a 512-byte boundary.	"512"	false	MEDIUM
camel.sink.endpoint.accessKey	Access key for the associated azure account name to be used for authentication with azure blob services	null	false	MEDIUM
camel.component.azure-storage-blob.autoDiscover Client	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	false	MEDIUM
camel.component.azure-storage-blob.blobName	The blob name, to consume specific blob from a container. However on producer, is only required for the operations on the blob level	null	false	MEDIUM
camel.component.azure-storage-blob.blobOffset	Set the blob offset for the upload or download operations, default is 0	0L	false	MEDIUM
camel.component.azure-storage-blob.blobType	The blob type in order to initiate the appropriate settings for each blob type One of: [blockblob] [appendblob] [pageblob]	"blockblob"	false	MEDIUM
camel.component.azure-storage-blob.closeStream AfterRead	Close the stream after read or keep it open, default is true	true	false	MEDIUM
camel.component.azure-storage-blob.configuration	The component configurations	null	false	MEDIUM
camel.component.azure-storage-blob.credentials	StorageSharedKeyCredential can be injected to create the azure client, this holds the important authentication information	null	false	MEDIUM
camel.component.azure-storage-blob.dataCount	How many bytes to include in the range. Must be greater than or equal to 0 if specified.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.azure-storage-blob.fileDir	The file directory where the downloaded blobs will be saved to, this can be used in both, producer and consumer	null	false	MEDIUM
camel.component.azure-storage-blob.maxResultsPerPage	Specifies the maximum number of blobs to return, including all BlobPrefix elements. If the request does not specify maxResultsPerPage or specifies a value greater than 5,000, the server will return up to 5,000 items.	null	false	MEDIUM
camel.component.azure-storage-blob.maxRetryRequests	Specifies the maximum number of additional HTTP Get requests that will be made while reading the data from a response body.	0	false	MEDIUM
camel.component.azure-storage-blob.prefix	Filters the results to return only blobs whose names begin with the specified prefix. May be null to return all blobs.	null	false	MEDIUM
camel.component.azure-storage-blob.regex	Filters the results to return only blobs whose names match the specified regular expression. May be null to return all if both prefix and regex are set, regex takes the priority and prefix is ignored.	null	false	MEDIUM
camel.component.azure-storage-blob.serviceClient	Client to a storage account. This client does not hold any state about a particular storage account but is instead a convenient way of sending off appropriate requests to the resource on the service. It may also be used to construct URLs to blobs and containers. This client contains operations on a service account. Operations on a container are available on BlobContainerClient through BlobServiceClient#getBlobContainerClient(String), and operations on a blob are available on BlobClient through BlobContainerClient#getBlobClient(String).	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.azure-storage-blob.timeout	An optional timeout value beyond which a RuntimeException will be raised.	null	false	MEDIUM
camel.component.azure-storage-blob.blobSequence Number	A user-controlled value that you can use to track requests. The value of the sequence number must be between 0 and 263 - 1.The default value is 0.	"0"	false	MEDIUM
camel.component.azure-storage-blob.blockListType	Specifies which type of blocks to return. One of: [committed] [uncommitted] [all]	"COMMIT TED"	false	MEDIUM
camel.component.azure-storage-blob.closeStream AfterWrite	Close the stream after write or keep it open, default is true	true	false	MEDIUM
camel.component.azure-storage-blob.commitBlockList Later	When is set to true, the staged blocks will not be committed directly.	true	false	MEDIUM
camel.component.azure-storage-blob.createAppend Blob	When is set to true, the append blocks will be created when committing append blocks.	true	false	MEDIUM
camel.component.azure-storage-blob.createPageBlob	When is set to true, the page blob will be created when uploading page blob.	true	false	MEDIUM
camel.component.azure-storage-blob.downloadLink Expiration	Override the default expiration (millis) of URL download link.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.azure-storage-blob.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	false	MEDIUM
camel.component.azure-storage-blob.operation	The blob operation that can be used with this component on the producer One of: [listBlobContainers] [createBlobContainer] [deleteBlobContainer] [listBlobs] [getBlob] [deleteBlob] [downloadBlobToFile] [downloadLink] [uploadBlockBlob] [stageBlockBlobList] [commitBlobBlockList] [getBlobBlockList] [createAppendBlob] [commitAppendBlob] [createPageBlob] [uploadPageBlob] [resizePageBlob] [clearPageBlob] [getPageBlobRanges]	"listBlobContainers"	false	MEDIUM
camel.component.azure-storage-blob.pageBlobSize	Specifies the maximum size for the page blob, up to 8 TB. The page blob size must be aligned to a 512-byte boundary.	"512"	false	MEDIUM
camel.component.azure-storage-blob.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.azure-storage-blob.accessKey	Access key for the associated azure account name to be used for authentication with azure blob services	null	false	MEDIUM

The camel-azure-storage-blob sink connector has no converters out of the box.

The camel-azure-storage-blob sink connector has no transforms out of the box.

The camel-azure-storage-blob sink connector has no aggregation strategies out of the box.

5.5.2. camel-azure-storage-queue-kafka-connector sink configuration

Connector Description: The azure-storage-queue component is used for storing and retrieving the messages to/from Azure Storage Queue using Azure SDK v12.

When using camel-azure-storage-queue-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-azure-storage-queue-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.azurestoragequeue.CamelAzurestoragequeueSinkC
onnector
```

The camel-azure-storage-queue sink connector supports 30 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.accountName	Azure account name to be used for authentication with azure queue services	null	false	MEDIUM
camel.sink.path.queueName	The queue resource name	null	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.serviceClient	Service client to a storage account to interact with the queue service. This client does not hold any state about a particular storage account but is instead a convenient way of sending off appropriate requests to the resource on the service. This client contains all the operations for interacting with a queue account in Azure Storage. Operations allowed by the client are creating, listing, and deleting queues, retrieving and updating properties of the account, and retrieving statistics of the account.	null	false	MEDIUM
camel.sink.endpoint.createQueue	When is set to true, the queue will be automatically created when sending messages to the queue.	true	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	false	MEDIUM
camel.sink.endpoint.operation	Queue service operation hint to the producer One of: [listQueues] [createQueue] [deleteQueue] [clearQueue] [sendMessage] [deleteMessage] [receiveMessages] [peekMessages] [updateMessage]	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.maxMessages	Maximum number of messages to get, if there are less messages exist in the queue than requested all the messages will be returned. If left empty only 1 message will be retrieved, the allowed range is 1 to 32 messages.	"1"	false	MEDIUM
camel.sink.endpoint.messageId	The ID of the message to be deleted or updated.	null	false	MEDIUM
camel.sink.endpoint.popReceipt	Unique identifier that must match for the message to be deleted or updated.	null	false	MEDIUM
camel.sink.endpoint.timeout	An optional timeout applied to the operation. If a response is not returned before the timeout concludes a RuntimeException will be thrown.	null	false	MEDIUM
camel.sink.endpoint.timeToLive	How long the message will stay alive in the queue. If unset the value will default to 7 days, if -1 is passed the message will not expire. The time to live must be -1 or any positive number. The format should be in this form: PnDTnHnMn.nS., e.g: PT20.345S — parses as 20.345 seconds, P2D — parses as 2 days However, in case you are using EndpointDsl/ComponentDsl, you can do something like Duration.ofSeconds() since these Java APIs are typesafe.	null	false	MEDIUM
camel.sink.endpoint.visibilityTimeout	The timeout period for how long the message is invisible in the queue. The timeout must be between 1 seconds and 7 days. The format should be in this form: PnDTnHnMn.nS., e.g: PT20.345S — parses as 20.345 seconds, P2D — parses as 2 days However, in case you are using EndpointDsl/ComponentDsl, you can do something like Duration.ofSeconds() since these Java APIs are typesafe.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.accessKey	Access key for the associated azure account name to be used for authentication with azure queue services	null	false	MEDIUM
camel.sink.endpoint.credentials	StorageSharedKeyCredential can be injected to create the azure client, this holds the important authentication information	null	false	MEDIUM
camel.component.azure-storage-queue.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	false	MEDIUM
camel.component.azure-storage-queue.configuration	The component configurations	null	false	MEDIUM
camel.component.azure-storage-queue.serviceClient	Service client to a storage account to interact with the queue service. This client does not hold any state about a particular storage account but is instead a convenient way of sending off appropriate requests to the resource on the service. This client contains all the operations for interacting with a queue account in Azure Storage. Operations allowed by the client are creating, listing, and deleting queues, retrieving and updating properties of the account, and retrieving statistics of the account.	null	false	MEDIUM
camel.component.azure-storage-queue.createQueue	When is set to true, the queue will be automatically created when sending messages to the queue.	true	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.azure-storage-queue.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	false	MEDIUM
camel.component.azure-storage-queue.operation	Queue service operation hint to the producer One of: [listQueues] [createQueue] [deleteQueue] [clearQueue] [sendMessage] [deleteMessage] [receiveMessages] [peekMessages] [updateMessage]	null	false	MEDIUM
camel.component.azure-storage-queue.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.azure-storage-queue.maxMessages	Maximum number of messages to get, if there are less messages exist in the queue than requested all the messages will be returned. If left empty only 1 message will be retrieved, the allowed range is 1 to 32 messages.	"1"	false	MEDIUM
camel.component.azure-storage-queue.messageId	The ID of the message to be deleted or updated.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.azure-storage-queue.popReceipt	Unique identifier that must match for the message to be deleted or updated.	null	false	MEDIUM
camel.component.azure-storage-queue.timeout	An optional timeout applied to the operation. If a response is not returned before the timeout concludes a RuntimeException will be thrown.	null	false	MEDIUM
camel.component.azure-storage-queue.timeToLive	How long the message will stay alive in the queue. If unset the value will default to 7 days, if -1 is passed the message will not expire. The time to live must be -1 or any positive number. The format should be in this form: PnDTnHnMn.nS., e.g: PT20.345S — parses as 20.345 seconds, P2D — parses as 2 days However, in case you are using EndpointDsl/ComponentDsl, you can do something like Duration.ofSeconds() since these Java APIs are typesafe.	null	false	MEDIUM
camel.component.azure-storage-queue.visibility Timeout	The timeout period for how long the message is invisible in the queue. The timeout must be between 1 seconds and 7 days. The format should be in this form: PnDTnHnMn.nS., e.g: PT20.345S — parses as 20.345 seconds, P2D — parses as 2 days However, in case you are using EndpointDsl/ComponentDsl, you can do something like Duration.ofSeconds() since these Java APIs are typesafe.	null	false	MEDIUM
camel.component.azure-storage-queue.accessKey	Access key for the associated azure account name to be used for authentication with azure queue services	null	false	MEDIUM
camel.component.azure-storage-queue.credentials	StorageSharedKeyCredential can be injected to create the azure client, this holds the important authentication information	null	false	MEDIUM

The camel-azure-storage-queue sink connector has no converters out of the box.

The camel-azure-storage-queue sink connector has no transforms out of the box.

The camel-azure-storage-queue sink connector has no aggregation strategies out of the box.

5.6. CASSANDRA QUERY LANUAGE

5.6.1. camel-cql-kafka-connector sink configuration

Connector Description: Integrate with Cassandra 2.0 using the CQL3 API (not the Thrift API). Based on Cassandra Java Driver provided by DataStax.

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 17 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.beanRef	beanRef is defined using bean:id	null	false	MEDIUM
camel.sink.path.hosts	Hostname(s) cassansdra server(s). Multiple hosts can be separated by comma.	null	false	MEDIUM
camel.sink.path.port	Port number of cassansdra server(s)	null	false	MEDIUM
camel.sink.path.keyspace	Keyspace to use	null	false	MEDIUM
camel.sink.endpoint.clusterName	Cluster name	null	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.cql	CQL query to perform. Can be overridden with the message header with key CamelCqlQuery.	null	false	MEDIUM
camel.sink.endpoint.datacenter	Datacenter to use	"datacenter1"	false	MEDIUM
camel.sink.endpoint.loadBalancingPolicyClass	To use a specific LoadBalancingPolicyClass	null	false	MEDIUM
camel.sink.endpoint.password	Password for session authentication	null	false	MEDIUM
camel.sink.endpoint.prepareStatements	Whether to use PreparedStatements or regular Statements	true	false	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	false	MEDIUM
camel.sink.endpoint.session	To use the Session instance (you would normally not use this option)	null	false	MEDIUM
camel.sink.endpoint.username	Username for session authentication	null	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.component.cql.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	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.

5.6.2. camel-cql-kafka-connector source configuration

Connector description: Integrate with Cassandra 2.0 using the CQL3 API (not the Thrift API). Based on Cassandra Java Driver provided by DataStax.

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 35 options, which are listed below.

Name	Description	Default	Required	Priority
camel.source.path.beanRef	beanRef is defined using bean:id	null	false	MEDIUM
camel.source.path.hosts	Hostname(s) cassansdra server(s). Multiple hosts can be separated by comma.	null	false	MEDIUM
camel.source.path.port	Port number of cassansdra server(s)	null	false	MEDIUM
camel.source.path.keyspace	Keyspace to use	null	false	MEDIUM
camel.source.endpoint.clusterName	Cluster name	null	false	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	false	MEDIUM
camel.source.endpoint.cql	CQL query to perform. Can be overridden with the message header with key CamelCqlQuery.	null	false	MEDIUM
camel.source.endpoint.datacenter	Datacenter to use	"datacenter1"	false	MEDIUM
camel.source.endpoint.loadBalancingPolicyClass	To use a specific LoadBalancingPolicyClass	null	false	MEDIUM
camel.source.endpoint.password	Password for session authentication	null	false	MEDIUM
camel.source.endpoint.prepareStatements	Whether to use PreparedStatements or regular Statements	true	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.session	To use the Session instance (you would normally not use this option)	null	false	MEDIUM
camel.source.endpoint.username	Username for session authentication	null	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	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	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	false	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	false	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	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.backoffIdleThreshold	The number of subsequent idle polls that should happen before the backoffMultipler should kick-in.	null	false	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	false	MEDIUM
camel.source.endpoint.delay	Milliseconds before the next poll.	500L	false	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	false	MEDIUM
camel.source.endpoint.initialDelay	Milliseconds before the first poll starts.	1000L	false	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	false	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"	false	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	false	MEDIUM

Name	Description	Default	Required	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"	false	MEDIUM
camel.source.endpoint.schedulerProperties	To configure additional properties when using a custom scheduler or any of the Quartz, Spring based scheduler.	null	false	MEDIUM
camel.source.endpoint.startScheduler	Whether the scheduler should be auto started.	true	false	MEDIUM
camel.source.endpoint.timeUnit	Time unit for initialDelay and delay options. One of: [NANOSECONDS] [MICROSECONDS] [MILLISECONDS] [SECONDS] [MINUTES] [HOURS] [DAYS]	"MILLISECONDS"	false	MEDIUM
camel.source.endpoint.useFixedDelay	Controls if fixed delay or fixed rate is used. See ScheduledExecutorService in JDK for details.	true	false	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	false	MEDIUM
camel.component.cql.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM

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

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

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

5.7. ELASTICSEARCH

5.7.1. camel-elasticsearch-rest-kafka-connector sink configuration

Connector Description: Send requests to with an ElasticSearch via REST API.

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 31 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.clusterName	Name of the cluster	null	true	HIGH
camel.sink.endpoint.connectionTimeout	The time in ms to wait before connection will timeout.	30000	false	MEDIUM
camel.sink.endpoint.disconnect	Disconnect after it finish calling the producer	false	false	MEDIUM
camel.sink.endpoint.enableSniffer	Enable automatically discover nodes from a running Elasticsearch cluster	false	false	MEDIUM
camel.sink.endpoint.enableSSL	Enable SSL	false	false	MEDIUM
camel.sink.endpoint.from	Starting index of the response.	null	false	MEDIUM
camel.sink.endpoint.hostAddresses	Comma separated list with ip:port formatted remote transport addresses to use.	null	true	HIGH

Name	Description	Default	Required	Priority
camel.sink.endpoint.indexName	The name of the index to act against	null	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	false	MEDIUM
camel.sink.endpoint.maxRetryTimeout	The time in ms before retry	30000	false	MEDIUM
camel.sink.endpoint.operation	What operation to perform One of: [Index] [Update] [Bulk] [BulkIndex] [GetById] [MultiGet] [MultiSearch] [Delete] [DeleteIndex] [Search] [Exists] [Ping]	null	false	MEDIUM
camel.sink.endpoint.scrollKeepAliveMs	Time in ms during which elasticsearch will keep search context alive	60000	false	MEDIUM
camel.sink.endpoint.size	Size of the response.	null	false	MEDIUM
camel.sink.endpoint.sniffAfterFailureDelay	The delay of a sniff execution scheduled after a failure (in milliseconds)	60000	false	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	false	MEDIUM
camel.sink.endpoint.socketTimeout	The timeout in ms to wait before the socket will timeout.	30000	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.useScroll	Enable scroll usage	false	false	MEDIUM
camel.sink.endpoint.waitForActiveShards	Index creation waits for the write consistency number of shards to be available	1	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	false	MEDIUM
camel.component.elasticsearch-rest.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	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	false	MEDIUM
camel.component.elasticsearch-rest.connectionTimeout	The time in ms to wait before connection will timeout.	30000	false	MEDIUM
camel.component.elasticsearch-rest.enableSniffer	Enable automatically discover nodes from a running Elasticsearch cluster	"false"	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.component.elasticsearch-rest.maxRetryTimeout	The time in ms before retry	30000	false	MEDIUM
camel.component.elasticsearch-rest.sniffAfterFailureDelay	The delay of a sniff execution scheduled after a failure (in milliseconds)	60000	false	MEDIUM
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	false	MEDIUM
camel.component.elasticsearch-rest.socketTimeout	The timeout in ms to wait before the socket will timeout.	30000	false	MEDIUM
camel.component.elasticsearch-rest.enableSSL	Enable SSL	"false"	false	MEDIUM
camel.component.elasticsearch-rest.password	Password for authenticate	null	false	MEDIUM
camel.component.elasticsearch-rest.user	Basic authenticate user	null	false	MEDIUM

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

The camel-elasticsearch-rest sink connector supports 0 transforms out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.elasticsearchrest.transformers.ConnectRecordValueToMapTransforms
```

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

5.8. FILE

5.8.1. camel-file-kafka-connector sink configuration

Connector Description: Read and write files.

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 26 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.directoryName	The starting directory	null	true	HIGH
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	false	MEDIUM

Name	Description	Default	Required	Priority
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 ${file.name.next} is supported as dynamic placeholders.	null	false	MEDIUM
camel.sink.endpoint.fileName	Use Expression such as File Language to dynamically set the filename. For consumers, it's used as a filename filter. For producers, it's used to evaluate the filename to write. If an expression is set, it take precedence over the CamelFileName header. (Note: The header itself can also be an Expression). The expression options support both String and Expression types. If the expression is a String type, it is always evaluated using the File Language. If the expression is an Expression type, the specified Expression type is used - this allows you, for instance, to use OGNL expressions. For the consumer, you can use it to filter filenames, so you can for instance consume today's file using the File Language syntax: mydata-${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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
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"	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	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	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	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	false	MEDIUM
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	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	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	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	false	MEDIUM
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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	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	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	false	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	false	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	false	MEDIUM
camel.sink.endpoint.synchronous	Sets whether synchronous processing should be strictly used	false	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.component.file.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	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.

5.9. HADOOP DISTRIBUTED FILE SYSTEM

5.9.1. camel-hdfs-kafka-connector sink configuration

Connector Description: Read and write from/to an HDFS filesystem using Hadoop 2.x.

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 30 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.hostName	HDFS host to use	null	true	HIGH
camel.sink.path.port	HDFS port to use	8020	false	MEDIUM
camel.sink.path.path	The directory path to use	null	true	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	false	MEDIUM
camel.sink.endpoint.fileSystemType	Set to LOCAL to not use HDFS but local java.io.File instead. One of: [LOCAL] [HDFS]	"HDFS"	false	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"	false	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"	false	MEDIUM
camel.sink.endpoint.namedNodes	A comma separated list of named nodes (e.g. srv11.example.com:8020,srv12.example.com:8020)	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.owner	The file owner must match this owner for the consumer to pickup the file. Otherwise the file is skipped.	null	false	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"	false	MEDIUM
camel.sink.endpoint.append	Append to existing file. Notice that not all HDFS file systems support the append option.	false	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	false	MEDIUM
camel.sink.endpoint.overwrite	Whether to overwrite existing files with the same name	true	false	MEDIUM
camel.sink.endpoint.blockSize	The size of the HDFS blocks	67108864L	false	MEDIUM
camel.sink.endpoint.bufferSize	The buffer size used by HDFS	4096	false	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	false	MEDIUM
camel.sink.endpoint.chunkSize	When reading a normal file, this is split into chunks producing a message per chunk.	4096	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.compressionCodec	The compression codec to use One of: [DEFAULT] [GZIP] [BZIP2]	"DEFAULT"	false	MEDIUM
camel.sink.endpoint.compressionType	The compression type to use (is default not in use) One of: [NONE] [RECORD] [BLOCK]	"NONE"	false	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"	false	MEDIUM
camel.sink.endpoint.readSuffix	Once the file has been read is renamed with this suffix to avoid to read it again.	"read"	false	MEDIUM
camel.sink.endpoint.replication	The HDFS replication factor	3	false	MEDIUM
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	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.sink.endpoint.kerberosUsername	The username used to authenticate with the kerberos nodes	null	false	MEDIUM
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	false	MEDIUM
camel.component.hdfs.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.hdfs.jAASConfiguration	To use the given configuration for security with JAAS.	null	false	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	false	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.

5.10. HYPERTEXT TRANSFER PROTOCOL

5.10.1. camel-http-kafka-connector sink configuration

Connector Description: Send requests to external HTTP servers using Apache HTTP Client 4.x.

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 78 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.httpUri	The url of the HTTP endpoint to call.	null	true	HIGH

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.sink.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to filter header to and from Camel message.	null	false	MEDIUM
camel.sink.endpoint.httpBinding	To use a custom HttpBinding to control the mapping between Camel message and HttpClient.	null	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	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	false	MEDIUM
camel.sink.endpoint.connectionClose	Specifies whether a Connection Close header must be added to HTTP Request. By default connectionClose is false.	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	false	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	false	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	false	MEDIUM
camel.sink.endpoint.ignoreResponseBody	If this option is true, The http producer won't read response body and cache the input stream	false	false	MEDIUM

Name	Description	Default	Required	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	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	false	MEDIUM
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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.sink.endpoint.cookieHandler	Configure a cookie handler to maintain a HTTP session	null	false	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	false	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	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	false	MEDIUM

Name	Description	Default	Required	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"	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	false	MEDIUM
camel.sink.endpoint.clientConnectionManager	To use a custom HttpClientConnectionManager to manage connections	null	false	MEDIUM
camel.sink.endpoint.connectionsPerRoute	The maximum number of connections per route.	20	false	MEDIUM
camel.sink.endpoint.httpClient	Sets a custom HttpClient to be used by the producer	null	false	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	false	MEDIUM
camel.sink.endpoint.httpClientOptions	To configure the HttpClient using the key/values from the Map.	null	false	MEDIUM
camel.sink.endpoint.httpContext	To use a custom HttpContext instance	null	false	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	false	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	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.sink.endpoint.maxTotalConnections	The maximum number of connections.	200	false	MEDIUM
camel.sink.endpoint.useSystemProperties	To use System Properties as fallback for configuration	false	false	MEDIUM
camel.sink.endpoint.proxyAuthDomain	Proxy authentication domain to use with NTML	null	false	MEDIUM
camel.sink.endpoint.proxyAuthHost	Proxy authentication host	null	false	MEDIUM
camel.sink.endpoint.proxyAuthMethod	Proxy authentication method to use One of: [Basic] [Digest] [NTLM]	null	false	MEDIUM
camel.sink.endpoint.proxyAuthNtHost	Proxy authentication domain (workstation name) to use with NTML	null	false	MEDIUM
camel.sink.endpoint.proxyAuthPassword	Proxy authentication password	null	false	MEDIUM
camel.sink.endpoint.proxyAuthPort	Proxy authentication port	null	false	MEDIUM
camel.sink.endpoint.proxyAuthScheme	Proxy authentication scheme to use One of: [http] [https]	null	false	MEDIUM
camel.sink.endpoint.proxyAuthUsername	Proxy authentication username	null	false	MEDIUM
camel.sink.endpoint.proxyHost	Proxy hostname to use	null	false	MEDIUM
camel.sink.endpoint.proxyPort	Proxy port to use	null	false	MEDIUM
camel.sink.endpoint.authDomain	Authentication domain to use with NTML	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.authenticationPreemptive	If this option is true, camel-http sends preemptive basic authentication to the server.	false	false	MEDIUM
camel.sink.endpoint.authHost	Authentication host to use with NTML	null	false	MEDIUM
camel.sink.endpoint.authMethod	Authentication methods allowed to use as a comma separated list of values Basic, Digest or NTLM.	null	false	MEDIUM
camel.sink.endpoint.authMethodPriority	Which authentication method to prioritize to use, either as Basic, Digest or NTLM. One of: [Basic] [Digest] [NTLM]	null	false	MEDIUM
camel.sink.endpoint.authPassword	Authentication password	null	false	MEDIUM
camel.sink.endpoint.authUsername	Authentication username	null	false	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	false	MEDIUM
camel.sink.endpoint.x509HostnameVerifier	To use a custom X509HostnameVerifier such as DefaultHostnameVerifier or NoopHostnameVerifier	null	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
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	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	false	MEDIUM
camel.component.http.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.component.http.connectionsPerRoute	The maximum number of connections per route.	20	false	MEDIUM
camel.component.http.connectionTimeToLive	The time for connection to live, the time unit is millisecond, the default value is always keep alive.	null	false	MEDIUM
camel.component.http.httpBinding	To use a custom HttpBinding to control the mapping between Camel message and HttpClient.	null	false	MEDIUM
camel.component.http.httpClientConfigurer	To use the custom HttpClientConfigurer to perform configuration of the HttpClient that will be used.	null	false	MEDIUM
camel.component.http.httpConfiguration	To use the shared HttpConfiguration as base configuration.	null	false	MEDIUM
camel.component.http.httpContext	To use a custom org.apache.http.protocol.HttpContext when executing requests.	null	false	MEDIUM
camel.component.http.maxTotalConnections	The maximum number of connections.	200	false	MEDIUM
camel.component.http.headerFilterStrategy	To use a custom org.apache.camel.spi.HeaderFilterStrategy to filter header to and from Camel message.	null	false	MEDIUM
camel.component.http.proxyAuthDomain	Proxy authentication domain to use	null	false	MEDIUM
camel.component.http.proxyAuthHost	Proxy authentication host	null	false	MEDIUM
camel.component.http.proxyAuthMethod	Proxy authentication method to use One of: [Basic] [Digest] [NTLM]	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.http.proxyAuthNtHost	Proxy authentication domain (workstation name) to use with NTML	null	false	MEDIUM
camel.component.http.proxyAuthPassword	Proxy authentication password	null	false	MEDIUM
camel.component.http.proxyAuthPort	Proxy authentication port	null	false	MEDIUM
camel.component.http.proxyAuthUsername	Proxy authentication username	null	false	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	false	MEDIUM
camel.component.http.useGlobalSslContextParameters	Enable usage of global SSL context parameters.	false	false	MEDIUM
camel.component.http.x509HostnameVerifier	To use a custom X509HostnameVerifier such as DefaultHostnameVerifier or NoopHostnameVerifier.	null	false	MEDIUM
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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	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	false	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.

5.11. JAVA DATABASE CONNECTIVITY

5.11.1. camel-jdbc-kafka-connector sink configuration

Connector Description: Access databases through SQL and JDBC.

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 17 options, which are listed below.

Name	Description	Default	Required	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	true	HIGH
camel.sink.endpoint.allowNamedParameters	Whether to allow using named parameters in the queries.	true	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	false	MEDIUM
camel.sink.endpoint.outputClass	Specify the full package and class name to use as conversion when outputType=SelectOne or SelectList.	null	false	MEDIUM
camel.sink.endpoint.outputType	Determines the output the producer should use. One of: [SelectOne] [SelectList] [StreamList]	"SelectList"	false	MEDIUM
camel.sink.endpoint.parameters	Optional parameters to the java.sql.Statement. For example to set maxRows, fetchSize etc.	null	false	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	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.sink.endpoint.transacted	Whether transactions are in use.	false	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	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	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
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	false	MEDIUM
camel.component.jdbc.dataSource	To use the DataSource instance instead of looking up the data source by name from the registry.	null	false	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	false	MEDIUM
camel.component.jdbc.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	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.

5.12. JAVA MESSAGE SERVICE

5.12.1. camel-sjms-kafka-connector sink configuration

Connector Description: Send and receive messages to/from a JMS Queue or Topic using plain JMS 1.x API.

When using camel-sjms-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-sjms-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.sjms.CamelSjmsSinkConnector
```

The camel-sjms sink connector supports 43 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.destinationType	The kind of destination to use One of: [queue] [topic]	"queue"	false	MEDIUM
camel.sink.path.destinationName	DestinationName is a JMS queue or topic name. By default, the destinationName is interpreted as a queue name.	null	true	HIGH
camel.sink.endpoint.acknowledgementMode	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"	false	MEDIUM
camel.sink.endpoint.connectionFactory	The connection factory to be use. A connection factory must be configured either on the component or endpoint.	null	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.sink.endpoint.replyTo	Provides an explicit ReplyTo destination (overrides any incoming value of Message.getJMSReplyTo() in consumer).	null	false	MEDIUM
camel.sink.endpoint.testConnectionOnStartup	Specifies whether to test the connection on startup. This ensures that when Camel starts that all the JMS consumers have a valid connection to the JMS broker. If a connection cannot be granted then Camel throws an exception on startup. This ensures that Camel is not started with failed connections. The JMS producers is tested as well.	false	false	MEDIUM
camel.sink.endpoint.deliveryMode	Specifies the delivery mode to be used. Possible values are those defined by javax.jms.DeliveryMode. NON_PERSISTENT = 1 and PERSISTENT = 2. One of: [1] [2]	null	false	MEDIUM
camel.sink.endpoint.deliveryPersistent	Specifies whether persistent delivery is used by default.	true	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.sink.endpoint.priority	Values greater than 1 specify the message priority when sending (where 1 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	false	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	false	MEDIUM
camel.sink.endpoint.replyToOverride	Provides an explicit ReplyTo destination in the JMS message, which overrides the setting of replyTo. It is useful if you want to forward the message to a remote Queue and receive the reply message from the ReplyTo destination.	null	false	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 or Exclusive. By default Camel will use temporary queues. However if replyTo has been configured, then Exclusive is used. One of: [Temporary] [Exclusive]	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.requestTimeout	The timeout for waiting for a reply when using the InOut Exchange Pattern (in milliseconds). The default is 20 seconds. You can include the header CamelJmsRequestTimeout to override this endpoint configured timeout value, and thus have per message individual timeout values. See also the requestTimeoutCheckerInterval option.	20000L	false	MEDIUM
camel.sink.endpoint.timeToLive	When sending messages, specifies the time-to-live of the message (in milliseconds).	-1L	false	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	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
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"	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	false	MEDIUM
camel.sink.endpoint.asyncStartListener	Whether to startup the consumer 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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.asyncStopListener	Whether to stop the consumer message listener asynchronously, when stopping a route.	false	false	MEDIUM
camel.sink.endpoint.destinationCreationStrategy	To use a custom DestinationCreationStrategy.	null	false	MEDIUM
camel.sink.endpoint.exceptionListener	Specifies the JMS Exception Listener that is to be notified of any underlying JMS exceptions.	null	false	MEDIUM
camel.sink.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to filter header to and from Camel message.	null	false	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	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.	null	false	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. See section about how mapping works below for more details.	true	false	MEDIUM

Name	Description	Default	Required	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	false	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	false	MEDIUM
camel.sink.endpoint.synchronous	Sets whether synchronous processing should be strictly used	false	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	false	MEDIUM
camel.sink.endpoint.transacted	Specifies whether to use transacted mode	false	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.sjms.connectionFactory	The connection factory to be use. A connection factory must be configured either on the component or endpoint.	null	false	MEDIUM
camel.component.sjms.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	false	MEDIUM
camel.component.sjms.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.sjms.destinationCreationStrategy	To use a custom DestinationCreationStrategy.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.sjms.jmsKeyFormatStrategy	Pluggable strategy for encoding and decoding JMS keys so they can be compliant with the JMS specification. Camel provides one implementation out of the box: default. The default strategy will safely marshal dots and hyphens (. and -). 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.	null	false	MEDIUM
camel.component.sjms.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	false	MEDIUM
camel.component.sjms.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	false	MEDIUM
camel.component.sjms.replyToOnTimeoutMaxConcurrent Consumers	Specifies the maximum number of concurrent consumers for continue routing when timeout occurred when using request/reply over JMS.	1	false	MEDIUM
camel.component.sjms.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	false	MEDIUM
camel.component.sjms.headerFilterStrategy	To use a custom org.apache.camel.spi.HeaderFilterStrategy to filter header to and from Camel message.	null	false	MEDIUM

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

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

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

5.12.2. camel-sjms-kafka-connector source configuration

Connector description: Send and receive messages to/from a JMS Queue or Topic using plain JMS 1.x API.

When using camel-sjms-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-sjms-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.sjms.CamelSjmsSourceConnector
```

The camel-sjms source connector supports 43 options, which are listed below.

Name	Description	Default	Required	Priority
camel.source.path.destinationType	The kind of destination to use One of: [queue] [topic]	"queue"	false	MEDIUM
camel.source.path.destinationName	DestinationName is a JMS queue or topic name. By default, the destinationName is interpreted as a queue name.	null	true	HIGH
camel.source.endpoint.acknowledgementMode	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"	false	MEDIUM
camel.source.endpoint.connectionFactory	The connection factory to be use. A connection factory must be configured either on the component or endpoint.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.disableReplyTo	Specifies whether Camel ignores the JMSReplyTo header in messages. If true, Camel does not send a reply back to the destination specified in the JMSReplyTo header. You can use this option if you want Camel to consume from a route and you do not want Camel to automatically send back a reply message because another component in your code handles the reply message. You can also use this option if you want to use Camel as a proxy between different message brokers and you want to route message from one system to another.	false	false	MEDIUM
camel.source.endpoint.replyTo	Provides an explicit ReplyTo destination (overrides any incoming value of Message.getJMSReplyTo() in consumer).	null	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.asyncConsumer	Whether the JmsConsumer processes the Exchange asynchronously. If enabled then the JmsConsumer may pickup the next message from the JMS queue, while the previous message is being processed asynchronously (by the Asynchronous Routing Engine). This means that messages may be processed not 100% strictly in order. If disabled (as default) then the Exchange is fully processed before the JmsConsumer will pickup the next message from the JMS queue. Note if transacted has been enabled, then asyncConsumer=true does not run asynchronously, as transaction must be executed synchronously (Camel 3.0 may support async transactions).	false	false	MEDIUM
camel.source.endpoint.autoStartup	Specifies whether the consumer container should auto-startup.	true	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	false	MEDIUM
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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.source.endpoint.durableSubscriptionName	The durable subscriber name for specifying durable topic subscriptions. The clientId option must be configured as well.	null	false	MEDIUM
camel.source.endpoint.replyToDeliveryPersistent	Specifies whether to use persistent delivery by default for replies.	true	false	MEDIUM
camel.source.endpoint.eagerLoadingOfProperties	Enables eager loading of JMS properties and payload as soon as a message is loaded which generally is inefficient as the JMS properties may not be required but sometimes can catch early any issues with the underlying JMS provider and the use of JMS properties. See also the option eagerPoisonBody.	false	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}"	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	false	MEDIUM
camel.source.endpoint.messageSelector	Sets the JMS Message selector syntax.	null	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	false	MEDIUM
camel.source.endpoint.asyncStartListener	Whether to startup the consumer 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	false	MEDIUM
camel.source.endpoint.asyncStopListener	Whether to stop the consumer message listener asynchronously, when stopping a route.	false	false	MEDIUM
camel.source.endpoint.destinationCreationStrategy	To use a custom DestinationCreationStrategy.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.exceptionListener	Specifies the JMS Exception Listener that is to be notified of any underlying JMS exceptions.	null	false	MEDIUM
camel.source.endpoint.headerFilterStrategy	To use a custom HeaderFilterStrategy to filter header to and from Camel message.	null	false	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	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.	null	false	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. See section about how mapping works below for more details.	true	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.source.endpoint.synchronous	Sets whether synchronous processing should be strictly used	false	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	false	MEDIUM
camel.source.endpoint.transacted	Specifies whether to use transacted mode	false	false	MEDIUM
camel.component.sjms.connectionFactory	The connection factory to be use. A connection factory must be configured either on the component or endpoint.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.sjms.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	false	MEDIUM
camel.component.sjms.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.sjms.destinationCreationStrategy	To use a custom DestinationCreationStrategy.	null	false	MEDIUM
camel.component.sjms.jmsKeyFormatStrategy	Pluggable strategy for encoding and decoding JMS keys so they can be compliant with the JMS specification. Camel provides one implementation out of the box: default. The default strategy will safely marshal dots and hyphens (. and -). 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.	null	false	MEDIUM
camel.component.sjms.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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.sjms.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	false	MEDIUM
camel.component.sjms.replyToOnTimeoutMaxConcurrent Consumers	Specifies the maximum number of concurrent consumers for continue routing when timeout occurred when using request/reply over JMS.	1	false	MEDIUM
camel.component.sjms.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	false	MEDIUM
camel.component.sjms.headerFilterStrategy	To use a custom org.apache.camel.spi.HeaderFilterStrategy to filter header to and from Camel message.	null	false	MEDIUM

The camel-sjms source connector has no converters out of the box.

The camel-sjms source connector has no transforms out of the box.

The camel-sjms source connector has no aggregation strategies out of the box.

5.13. MONGODB

5.13.1. camel-mongodb-kafka-connector sink configuration

Connector Description: Perform operations on MongoDB documents and collections.

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 24 options, which are listed below.

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

Name	Description	Default	Required	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	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	false	MEDIUM
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	false	MEDIUM

Name	Description	Default	Required	Priority
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"	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"	false	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	false	MEDIUM
camel.sink.endpoint.streamFilter	Filter condition for change streams consumer.	null	false	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	false	MEDIUM

Name	Description	Default	Required	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	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	false	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	false	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	false	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	false	MEDIUM
camel.component.mongodb.mongoConnection	Shared client used for connection. All endpoints generated from the component will share this connection client.	null	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.component.mongodb.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	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.

5.13.2. camel-mongodb-kafka-connector source configuration

Connector description: Perform operations on MongoDB documents and collections.

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 27 options, which are listed below.

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

Name	Description	Default	Required	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	false	MEDIUM
camel.source.endpoint.consumerType	Consumer type.	null	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	false	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	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	false	MEDIUM

Name	Description	Default	Required	Priority
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	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"	false	MEDIUM
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"	false	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	false	MEDIUM
camel.source.endpoint.streamFilter	Filter condition for change streams consumer.	null	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	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	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	false	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	false	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	false	MEDIUM
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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.mongodb.mongoConnection	Shared client used for connection. All endpoints generated from the component will share this connection client.	null	false	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	false	MEDIUM
camel.component.mongodb.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM

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

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

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

5.14. RABBITMQ

5.14.1. camel-rabbitmq-kafka-connector sink configuration

Connector Description: Send and receive messages from RabbitMQ instances.

When using camel-rabbitmq-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-rabbitmq-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.rabbitmq.CamelRabbitmqSinkConnector
```

The camel-rabbitmq sink connector supports 100 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.exchangeName	The exchange name determines the exchange to which the produced messages will be sent to. In the case of consumers, the exchange name determines the exchange the queue will be bound to.	null	true	HIGH
camel.sink.endpoint.addresses	If this option is set, camel-rabbitmq will try to create connection based on the setting of option addresses. The addresses value is a string which looks like server1:12345, server2:12345	null	false	MEDIUM
camel.sink.endpoint.autoDelete	If it is true, the exchange will be deleted when it is no longer in use	true	false	MEDIUM
camel.sink.endpoint.automaticRecoveryEnabled	Enables connection automatic recovery (uses connection implementation that performs automatic recovery when existing connection has failures)	"true"	false	MEDIUM
camel.sink.endpoint.connectionFactory	To use a custom RabbitMQ connection factory. When this option is set, all connection options (connectionTimeout, requestedChannelMax...) set on URI are not used	null	false	MEDIUM
camel.sink.endpoint.deadLetterExchange	The name of the dead letter exchange	null	false	MEDIUM
camel.sink.endpoint.deadLetterExchangeType	The type of the dead letter exchange One of: [direct] [fanout] [headers] [topic]	"direct"	false	MEDIUM
camel.sink.endpoint.deadLetterQueue	The name of the dead letter queue	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.deadLetterRoutingKey	The routing key for the dead letter exchange	null	false	MEDIUM
camel.sink.endpoint.declare	If the option is true, camel declare the exchange and queue name and bind them together. If the option is false, camel won’t declare the exchange and queue name on the server.	true	false	MEDIUM
camel.sink.endpoint.durable	If we are declaring a durable exchange (the exchange will survive a server restart)	true	false	MEDIUM
camel.sink.endpoint.exchangeType	The exchange type such as direct or topic. One of: [direct] [fanout] [headers] [topic]	"direct"	false	MEDIUM
camel.sink.endpoint.exclusive	Exclusive queues may only be accessed by the current connection, and are deleted when that connection closes.	false	false	MEDIUM
camel.sink.endpoint.hostname	The hostname of the running rabbitmq instance or cluster.	null	false	MEDIUM
camel.sink.endpoint.passive	Passive queues depend on the queue already to be available at RabbitMQ.	false	false	MEDIUM
camel.sink.endpoint.portNumber	Port number for the host with the running rabbitmq instance or cluster. Default value is 5672.	null	false	MEDIUM
camel.sink.endpoint.queue	The queue to receive messages from	null	false	MEDIUM
camel.sink.endpoint.routingKey	The routing key to use when binding a consumer queue to the exchange. For producer routing keys, you set the header rabbitmq.ROUTING_KEY.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.skipDlqDeclare	If true the producer will not declare and bind a dead letter queue. This can be used if you have also DLQ rabbitmq consumer and you want to avoid argument clashing between Producer and Consumer. This option have no effect, if DLQ configured (deadLetterExchange option is not set).	false	false	MEDIUM
camel.sink.endpoint.skipExchangeDeclare	This can be used if we need to declare the queue but not the exchange	false	false	MEDIUM
camel.sink.endpoint.skipQueueBind	If true the queue will not be bound to the exchange after declaring it	false	false	MEDIUM
camel.sink.endpoint.skipQueueDeclare	If true the producer will not declare and bind a queue. This can be used for directing messages via an existing routing key.	false	false	MEDIUM
camel.sink.endpoint.vhost	The vhost for the channel	"/"	false	MEDIUM
camel.sink.endpoint.additionalHeaders	Map of additional headers. These headers will be set only when the 'allowCustomHeaders' is set to true	null	false	MEDIUM
camel.sink.endpoint.additionalProperties	Map of additional properties. These are standard RabbitMQ properties as defined in com.rabbitmq.client.AMQP.BasicProperties. The map keys should be from org.apache.camel.component.rabbitmq.RabbitMQConstants. Any other keys will be ignored.	null	false	MEDIUM
camel.sink.endpoint.allowCustomHeaders	Allow pass custom values to header	false	false	MEDIUM
camel.sink.endpoint.allowNullHeaders	Allow pass null values to header	false	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.bridgeEndpoint	If the bridgeEndpoint is true, the producer will ignore the message header of rabbitmq.EXCHANGE_NAME and rabbitmq.ROUTING_KEY	false	false	MEDIUM
camel.sink.endpoint.channelPoolMaxSize	Get maximum number of opened channel in pool	10	false	MEDIUM
camel.sink.endpoint.channelPoolMaxWait	Set the maximum number of milliseconds to wait for a channel from the pool	1000L	false	MEDIUM
camel.sink.endpoint.guaranteedDeliveries	When true, an exception will be thrown when the message cannot be delivered (basic.return) and the message is marked as mandatory. PublisherAcknowledgement will also be activated in this case. See also publisher acknowledgements - When will messages be confirmed.	false	false	MEDIUM
camel.sink.endpoint.immediate	This flag tells the server how to react if the message cannot be routed to a queue consumer immediately. If this flag is set, the server will return an undeliverable message with a Return method. If this flag is zero, the server will queue the message, but with no guarantee that it will ever be consumed. If the header is present rabbitmq.IMMEDIATE it will override this option.	false	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.sink.endpoint.mandatory	This flag tells the server how to react if the message cannot be routed to a queue. If this flag is set, the server will return an unroutable message with a Return method. If this flag is zero, the server silently drops the message. If the header is present rabbitmq.MANDATORY it will override this option.	false	false	MEDIUM
camel.sink.endpoint.publisherAcknowledgements	When true, the message will be published with publisher acknowledgements turned on	false	false	MEDIUM
camel.sink.endpoint.publisherAcknowledgementsTimeout	The amount of time in milliseconds to wait for a basic.ack response from RabbitMQ server	null	false	MEDIUM
camel.sink.endpoint.allowMessageBodySerialization	Whether to allow Java serialization of the message body or not. If this value is true, the message body will be serialized on the producer side using Java serialization, if no type converter can handle the message body. On the consumer side, it will deserialize the message body if this value is true and the message contains a CamelSerialize header. Setting this value to true may introduce a security vulnerability as it allows an attacker to attempt to deserialize to a gadget object which could result in a RCE or other security vulnerability.	false	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.args	Specify arguments for configuring the different RabbitMQ concepts, a different prefix is required for each: Exchange: arg.exchange. Queue: arg.queue. Binding: arg.binding. DLQ: arg.dlq.queue. DLQ binding: arg.dlq.binding. For example to declare a queue with message ttl argument: http://localhost:5672/exchange/queueargs=arg.queue.x-message-ttl=60000	null	false	MEDIUM
camel.sink.endpoint.clientProperties	Connection client properties (client info used in negotiating with the server)	null	false	MEDIUM
camel.sink.endpoint.connectionFactoryExceptio n Handler	Custom rabbitmq ExceptionHandler for ConnectionFactory	null	false	MEDIUM
camel.sink.endpoint.connectionTimeout	Connection timeout	60000	false	MEDIUM
camel.sink.endpoint.networkRecoveryInterval	Network recovery interval in milliseconds (interval used when recovering from network failure)	"5000"	false	MEDIUM
camel.sink.endpoint.requestedChannelMax	Connection requested channel max (max number of channels offered)	2047	false	MEDIUM
camel.sink.endpoint.requestedFrameMax	Connection requested frame max (max size of frame offered)	0	false	MEDIUM
camel.sink.endpoint.requestedHeartbeat	Connection requested heartbeat (heart-beat in seconds offered)	60	false	MEDIUM
camel.sink.endpoint.requestTimeout	Set timeout for waiting for a reply when using the InOut Exchange Pattern (in milliseconds)	20000L	false	MEDIUM
camel.sink.endpoint.requestTimeoutCheckerInterval	Set requestTimeoutCheckerInterval for inOut exchange	1000L	false	MEDIUM
camel.sink.endpoint.topologyRecoveryEnabled	Enables connection topology recovery (should topology recovery be performed)	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.transferException	When true and an inOut Exchange failed on the consumer side send the caused Exception back in the response	false	false	MEDIUM
camel.sink.endpoint.password	Password for authenticated access	"guest"	false	MEDIUM
camel.sink.endpoint.sslProtocol	Enables SSL on connection, accepted value are true, TLS and 'SSLv3	null	false	MEDIUM
camel.sink.endpoint.trustManager	Configure SSL trust manager, SSL should be enabled for this option to be effective	null	false	MEDIUM
camel.sink.endpoint.username	Username in case of authenticated access	"guest"	false	MEDIUM
camel.component.rabbitmq.addresses	If this option is set, camel-rabbitmq will try to create connection based on the setting of option addresses. The addresses value is a string which looks like server1:12345, server2:12345	null	false	MEDIUM
camel.component.rabbitmq.autoDelete	If it is true, the exchange will be deleted when it is no longer in use	true	false	MEDIUM
camel.component.rabbitmq.connectionFactory	To use a custom RabbitMQ connection factory. When this option is set, all connection options (connectionTimeout, requestedChannelMax...) set on URI are not used	null	false	MEDIUM
camel.component.rabbitmq.deadLetterExchange	The name of the dead letter exchange	null	false	MEDIUM
camel.component.rabbitmq.deadLetterExchangeType	The type of the dead letter exchange One of: [direct] [fanout] [headers] [topic]	"direct"	false	MEDIUM
camel.component.rabbitmq.deadLetterQueue	The name of the dead letter queue	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.rabbitmq.deadLetterRoutingKey	The routing key for the dead letter exchange	null	false	MEDIUM
camel.component.rabbitmq.declare	If the option is true, camel declare the exchange and queue name and bind them together. If the option is false, camel won't declare the exchange and queue name on the server.	true	false	MEDIUM
camel.component.rabbitmq.durable	If we are declaring a durable exchange (the exchange will survive a server restart)	true	false	MEDIUM
camel.component.rabbitmq.exclusive	Exclusive queues may only be accessed by the current connection, and are deleted when that connection closes.	false	false	MEDIUM
camel.component.rabbitmq.hostname	The hostname of the running RabbitMQ instance or cluster.	null	false	MEDIUM
camel.component.rabbitmq.passive	Passive queues depend on the queue already to be available at RabbitMQ.	false	false	MEDIUM
camel.component.rabbitmq.portNumber	Port number for the host with the running rabbitmq instance or cluster.	5672	false	MEDIUM
camel.component.rabbitmq.skipExchangeDeclare	This can be used if we need to declare the queue but not the exchange	false	false	MEDIUM
camel.component.rabbitmq.skipQueueBind	If true the queue will not be bound to the exchange after declaring it	false	false	MEDIUM
camel.component.rabbitmq.skipQueueDeclare	If true the producer will not declare and bind a queue. This can be used for directing messages via an existing routing key.	false	false	MEDIUM
camel.component.rabbitmq.vhost	The vhost for the channel	"/"	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.rabbitmq.additionalHeaders	Map of additional headers. These headers will be set only when the 'allowCustomHeaders' is set to true	null	false	MEDIUM
camel.component.rabbitmq.additionalProperties	Map of additional properties. These are standard RabbitMQ properties as defined in com.rabbitmq.client.AMQP.BasicProperties The map keys should be from org.apache.camel.component.rabbitmq.RabbitMQConstants. Any other keys will be ignored. When the message already contains these headers they will be given precedence over these properties.	null	false	MEDIUM
camel.component.rabbitmq.allowNullHeaders	Allow pass null values to header	false	false	MEDIUM
camel.component.rabbitmq.channelPoolMaxSize	Get maximum number of opened channel in pool	10	false	MEDIUM
camel.component.rabbitmq.channelPoolMaxWait	Set the maximum number of milliseconds to wait for a channel from the pool	1000L	false	MEDIUM
camel.component.rabbitmq.guaranteedDeliveries	When true, an exception will be thrown when the message cannot be delivered (basic.return) and the message is marked as mandatory. PublisherAcknowledgement will also be activated in this case. See also publisher acknowledgements - When will messages be confirmed.	false	false	MEDIUM
camel.component.rabbitmq.immediate	This flag tells the server how to react if the message cannot be routed to a queue consumer immediately. If this flag is set, the server will return an undeliverable message with a Return method. If this flag is zero, the server will queue the message, but with no guarantee that it will ever be consumed. If the header is present rabbitmq.IMMEDIATE it will override this option.	false	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.rabbitmq.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	false	MEDIUM
camel.component.rabbitmq.mandatory	This flag tells the server how to react if the message cannot be routed to a queue. If this flag is set, the server will return an unroutable message with a Return method. If this flag is zero, the server silently drops the message. If the header is present rabbitmq.MANDATORY it will override this option.	false	false	MEDIUM
camel.component.rabbitmq.publisherAcknowledgements	When true, the message will be published with publisher acknowledgements turned on	false	false	MEDIUM
camel.component.rabbitmq.publisherAcknowledgements Timeout	The amount of time in milliseconds to wait for a basic.ack response from RabbitMQ server	null	false	MEDIUM
camel.component.rabbitmq.args	Specify arguments for configuring the different RabbitMQ concepts, a different prefix is required for each: Exchange: arg.exchange. Queue: arg.queue. Binding: arg.binding. DLQ: arg.dlq.queue. DLQ Binding: arg.dlq.binding. For example to declare a queue with message ttl argument: http://localhost:5672/exchange/queueargs=arg.queue.x-message-ttl=60000	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.rabbitmq.autoDetectConnection Factory	Whether to auto-detect looking up RabbitMQ connection factory from the registry. When enabled and a single instance of the connection factory is found then it will be used. An explicit connection factory can be configured on the component or endpoint level which takes precedence.	true	false	MEDIUM
camel.component.rabbitmq.automaticRecoveryEnabled	Enables connection automatic recovery (uses connection implementation that performs automatic recovery when connection shutdown is not initiated by the application)	null	false	MEDIUM
camel.component.rabbitmq.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.rabbitmq.clientProperties	Connection client properties (client info used in negotiating with the server)	null	false	MEDIUM
camel.component.rabbitmq.connectionFactory ExceptionHandler	Custom rabbitmq ExceptionHandler for ConnectionFactory	null	false	MEDIUM
camel.component.rabbitmq.connectionTimeout	Connection timeout	60000	false	MEDIUM
camel.component.rabbitmq.networkRecoveryInterval	Network recovery interval in milliseconds (interval used when recovering from network failure)	"5000"	false	MEDIUM
camel.component.rabbitmq.requestedChannelMax	Connection requested channel max (max number of channels offered)	2047	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.rabbitmq.requestedFrameMax	Connection requested frame max (max size of frame offered)	0	false	MEDIUM
camel.component.rabbitmq.requestedHeartbeat	Connection requested heartbeat (heart-beat in seconds offered)	60	false	MEDIUM
camel.component.rabbitmq.requestTimeout	Set timeout for waiting for a reply when using the InOut Exchange Pattern (in milliseconds)	20000L	false	MEDIUM
camel.component.rabbitmq.requestTimeoutChecker Interval	Set requestTimeoutCheckerInterval for inOut exchange	1000L	false	MEDIUM
camel.component.rabbitmq.topologyRecoveryEnabled	Enables connection topology recovery (should topology recovery be performed)	null	false	MEDIUM
camel.component.rabbitmq.transferException	When true and an inOut Exchange failed on the consumer side send the caused Exception back in the response	false	false	MEDIUM
camel.component.rabbitmq.password	Password for authenticated access	"guest"	false	MEDIUM
camel.component.rabbitmq.sslProtocol	Enables SSL on connection, accepted value are true, TLS and 'SSLv3	null	false	MEDIUM
camel.component.rabbitmq.trustManager	Configure SSL trust manager, SSL should be enabled for this option to be effective	null	false	MEDIUM
camel.component.rabbitmq.username	Username in case of authenticated access	"guest"	false	MEDIUM

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

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

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

5.14.2. camel-rabbitmq-kafka-connector source configuration

Connector description: Send and receive messages from RabbitMQ instances.

When using camel-rabbitmq-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-rabbitmq-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.rabbitmq.CamelRabbitmqSourceConnector
```

The camel-rabbitmq source connector supports 97 options, which are listed below.

Name	Description	Default	Required	Priority
camel.source.path.exchangeName	The exchange name determines the exchange to which the produced messages will be sent to. In the case of consumers, the exchange name determines the exchange the queue will be bound to.	null	true	HIGH
camel.source.endpoint.addresses	If this option is set, camel-rabbitmq will try to create connection based on the setting of option addresses. The addresses value is a string which looks like server1:12345, server2:12345	null	false	MEDIUM
camel.source.endpoint.autoDelete	If it is true, the exchange will be deleted when it is no longer in use	true	false	MEDIUM
camel.source.endpoint.automaticRecoveryEnabled	Enables connection automatic recovery (uses connection implementation that performs automatic recovery when existing connection has failures)	"true"	false	MEDIUM
camel.source.endpoint.connectionFactory	To use a custom RabbitMQ connection factory. When this option is set, all connection options (connectionTimeout, requestedChannelMax...) set on URI are not used	null	false	MEDIUM
camel.source.endpoint.deadLetterExchange	The name of the dead letter exchange	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.deadLetterExchangeType	The type of the dead letter exchange One of: [direct] [fanout] [headers] [topic]	"direct"	false	MEDIUM
camel.source.endpoint.deadLetterQueue	The name of the dead letter queue	null	false	MEDIUM
camel.source.endpoint.deadLetterRoutingKey	The routing key for the dead letter exchange	null	false	MEDIUM
camel.source.endpoint.declare	If the option is true, camel declare the exchange and queue name and bind them together. If the option is false, camel won't declare the exchange and queue name on the server.	true	false	MEDIUM
camel.source.endpoint.durable	If we are declaring a durable exchange (the exchange will survive a server restart)	true	false	MEDIUM
camel.source.endpoint.exchangeType	The exchange type such as direct or topic. One of: [direct] [fanout] [headers] [topic]	"direct"	false	MEDIUM
camel.source.endpoint.exclusive	Exclusive queues may only be accessed by the current connection, and are deleted when that connection closes.	false	false	MEDIUM
camel.source.endpoint.hostname	The hostname of the running rabbitmq instance or cluster.	null	false	MEDIUM
camel.source.endpoint.passive	Passive queues depend on the queue already to be available at RabbitMQ.	false	false	MEDIUM
camel.source.endpoint.portNumber	Port number for the host with the running rabbitmq instance or cluster. Default value is 5672.	null	false	MEDIUM
camel.source.endpoint.queue	The queue to receive messages from	null	false	MEDIUM
camel.source.endpoint.routingKey	The routing key to use when binding a consumer queue to the exchange. For producer routing keys, you set the header rabbitmq.ROUTING_KEY.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.skipDlqDeclare	If true the producer will not declare and bind a dead letter queue. This can be used if you have also DLQ rabbitmq consumer and you want to avoid argument clashing between Producer and Consumer. This option have no effect, if DLQ configured (deadLetterExchange option is not set).	false	false	MEDIUM
camel.source.endpoint.skipExchangeDeclare	This can be used if we need to declare the queue but not the exchange	false	false	MEDIUM
camel.source.endpoint.skipQueueBind	If true the queue will not be bound to the exchange after declaring it	false	false	MEDIUM
camel.source.endpoint.skipQueueDeclare	If true the producer will not declare and bind a queue. This can be used for directing messages via an existing routing key.	false	false	MEDIUM
camel.source.endpoint.vhost	The vhost for the channel	"/"	false	MEDIUM
camel.source.endpoint.autoAck	If messages should be auto acknowledged	true	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	false	MEDIUM
camel.source.endpoint.concurrentConsumers	Number of concurrent consumers when consuming from broker. (eg similar as to the same option for the JMS component).	1	false	MEDIUM
camel.source.endpoint.consumerTag	Specify a client-generated consumer tag to establish context when invoking the consume operation	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.exclusiveConsumer	Request exclusive access to the queue (meaning only this consumer can access the queue). This is useful when you want a long-lived shared queue to be temporarily accessible by just one consumer.	false	false	MEDIUM
camel.source.endpoint.prefetchCount	The maximum number of messages that the server will deliver, 0 if unlimited. You need to specify the option of prefetchSize, prefetchCount, prefetchGlobal at the same time	null	false	MEDIUM
camel.source.endpoint.prefetchEnabled	Enables the quality of service on the RabbitMQConsumer side. You need to specify the option of prefetchSize, prefetchCount, prefetchGlobal at the same time	false	false	MEDIUM
camel.source.endpoint.prefetchGlobal	If the settings should be applied to the entire channel rather than each consumer You need to specify the option of prefetchSize, prefetchCount, prefetchGlobal at the same time	false	false	MEDIUM
camel.source.endpoint.prefetchSize	The maximum amount of content (measured in octets) that the server will deliver, 0 if unlimited. You need to specify the option of prefetchSize, prefetchCount, prefetchGlobal at the same time	null	false	MEDIUM
camel.source.endpoint.reQueue	This is used by the consumer to control rejection of the message. When the consumer is complete processing the exchange, and if the exchange failed, then the consumer is going to reject the message from the RabbitMQ broker. If the header CamelRabbitmqRequeue is present then the value of the header will be used, otherwise this endpoint value is used as fallback. If the value is false (by default) then the message is discarded/dead-lettered. If the value is true, then the message is re-queued.	false	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	false	MEDIUM
camel.source.endpoint.threadPoolSize	The consumer uses a Thread Pool Executor with a fixed number of threads. This setting allows you to set that number of threads.	10	false	MEDIUM
camel.source.endpoint.allowMessageBody Serialization	Whether to allow Java serialization of the message body or not. If this value is true, the message body will be serialized on the producer side using Java serialization, if no type converter can handle the message body. On the consumer side, it will deserialize the message body if this value is true and the message contains a CamelSerialize header. Setting this value to true may introduce a security vulnerability as it allows an attacker to attempt to deserialize to a gadget object which could result in a RCE or other security vulnerability.	false	false	MEDIUM
camel.source.endpoint.args	Specify arguments for configuring the different RabbitMQ concepts, a different prefix is required for each: Exchange: arg.exchange. Queue: arg.queue. Binding: arg.binding. DLQ: arg.dlq.queue. DLQ binding: arg.dlq.binding. For example to declare a queue with message ttl argument: http://localhost:5672/exchange/queueargs=arg.queue.x-message-ttl=60000	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.clientProperties	Connection client properties (client info used in negotiating with the server)	null	false	MEDIUM
camel.source.endpoint.connectionFactoryExcept ion Handler	Custom rabbitmq ExceptionHandler for ConnectionFactory	null	false	MEDIUM
camel.source.endpoint.connectionTimeout	Connection timeout	60000	false	MEDIUM
camel.source.endpoint.networkRecoveryInterval	Network recovery interval in milliseconds (interval used when recovering from network failure)	"5000"	false	MEDIUM
camel.source.endpoint.requestedChannelMax	Connection requested channel max (max number of channels offered)	2047	false	MEDIUM
camel.source.endpoint.requestedFrameMax	Connection requested frame max (max size of frame offered)	0	false	MEDIUM
camel.source.endpoint.requestedHeartbeat	Connection requested heartbeat (heart-beat in seconds offered)	60	false	MEDIUM
camel.source.endpoint.requestTimeout	Set timeout for waiting for a reply when using the InOut Exchange Pattern (in milliseconds)	20000L	false	MEDIUM
camel.source.endpoint.requestTimeoutChecker Interval	Set requestTimeoutCheckerInterval for inOut exchange	1000L	false	MEDIUM
camel.source.endpoint.topologyRecoveryEnabled	Enables connection topology recovery (should topology recovery be performed)	null	false	MEDIUM
camel.source.endpoint.transferException	When true and an inOut Exchange failed on the consumer side send the caused Exception back in the response	false	false	MEDIUM
camel.source.endpoint.password	Password for authenticated access	"guest"	false	MEDIUM
camel.source.endpoint.sslProtocol	Enables SSL on connection, accepted value are true, TLS and 'SSLv3	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.trustManager	Configure SSL trust manager, SSL should be enabled for this option to be effective	null	false	MEDIUM
camel.source.endpoint.username	Username in case of authenticated access	"guest"	false	MEDIUM
camel.component.rabbitmq.addresses	If this option is set, camel-rabbitmq will try to create connection based on the setting of option addresses. The addresses value is a string which looks like server1:12345, server2:12345	null	false	MEDIUM
camel.component.rabbitmq.autoDelete	If it is true, the exchange will be deleted when it is no longer in use	true	false	MEDIUM
camel.component.rabbitmq.connectionFactory	To use a custom RabbitMQ connection factory. When this option is set, all connection options (connectionTimeout, requestedChannelMax...) set on URI are not used	null	false	MEDIUM
camel.component.rabbitmq.deadLetterExchange	The name of the dead letter exchange	null	false	MEDIUM
camel.component.rabbitmq.deadLetterExchangeType	The type of the dead letter exchange One of: [direct] [fanout] [headers] [topic]	"direct"	false	MEDIUM
camel.component.rabbitmq.deadLetterQueue	The name of the dead letter queue	null	false	MEDIUM
camel.component.rabbitmq.deadLetterRoutingKey	The routing key for the dead letter exchange	null	false	MEDIUM
camel.component.rabbitmq.declare	If the option is true, camel declare the exchange and queue name and bind them together. If the option is false, camel won’t declare the exchange and queue name on the server.	true	false	MEDIUM
camel.component.rabbitmq.durable	If we are declaring a durable exchange (the exchange will survive a server restart)	true	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.rabbitmq.exclusive	Exclusive queues may only be accessed by the current connection, and are deleted when that connection closes.	false	false	MEDIUM
camel.component.rabbitmq.hostname	The hostname of the running RabbitMQ instance or cluster.	null	false	MEDIUM
camel.component.rabbitmq.passive	Passive queues depend on the queue already to be available at RabbitMQ.	false	false	MEDIUM
camel.component.rabbitmq.portNumber	Port number for the host with the running rabbitmq instance or cluster.	5672	false	MEDIUM
camel.component.rabbitmq.skipExchangeDeclare	This can be used if we need to declare the queue but not the exchange	false	false	MEDIUM
camel.component.rabbitmq.skipQueueBind	If true the queue will not be bound to the exchange after declaring it	false	false	MEDIUM
camel.component.rabbitmq.skipQueueDeclare	If true the producer will not declare and bind a queue. This can be used for directing messages via an existing routing key.	false	false	MEDIUM
camel.component.rabbitmq.vhost	The vhost for the channel	"/"	false	MEDIUM
camel.component.rabbitmq.autoAck	If messages should be auto acknowledged	true	false	MEDIUM
camel.component.rabbitmq.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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.rabbitmq.exclusiveConsumer	Request exclusive access to the queue (meaning only this consumer can access the queue). This is useful when you want a long-lived shared queue to be temporarily accessible by just one consumer.	false	false	MEDIUM
camel.component.rabbitmq.prefetchCount	The maximum number of messages that the server will deliver, 0 if unlimited. You need to specify the option of prefetchSize, prefetchCount, prefetchGlobal at the same time	null	false	MEDIUM
camel.component.rabbitmq.prefetchEnabled	Enables the quality of service on the RabbitMQConsumer side. You need to specify the option of prefetchSize, prefetchCount, prefetchGlobal at the same time	false	false	MEDIUM
camel.component.rabbitmq.prefetchGlobal	If the settings should be applied to the entire channel rather than each consumer You need to specify the option of prefetchSize, prefetchCount, prefetchGlobal at the same time	false	false	MEDIUM
camel.component.rabbitmq.prefetchSize	The maximum amount of content (measured in octets) that the server will deliver, 0 if unlimited. You need to specify the option of prefetchSize, prefetchCount, prefetchGlobal at the same time	null	false	MEDIUM
camel.component.rabbitmq.threadPoolSize	The consumer uses a Thread Pool Executor with a fixed number of threads. This setting allows you to set that number of threads.	10	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.rabbitmq.args	Specify arguments for configuring the different RabbitMQ concepts, a different prefix is required for each: Exchange: arg.exchange. Queue: arg.queue. Binding: arg.binding. DLQ: arg.dlq.queue. DLQ Binding: arg.dlq.binding. For example to declare a queue with message ttl argument: http://localhost:5672/exchange/queueargs=arg.queue.x-message-ttl=60000	null	false	MEDIUM
camel.component.rabbitmq.autoDetectConnection Factory	Whether to auto-detect looking up RabbitMQ connection factory from the registry. When enabled and a single instance of the connection factory is found then it will be used. An explicit connection factory can be configured on the component or endpoint level which takes precedence.	true	false	MEDIUM
camel.component.rabbitmq.automaticRecoveryEnabled	Enables connection automatic recovery (uses connection implementation that performs automatic recovery when connection shutdown is not initiated by the application)	null	false	MEDIUM
camel.component.rabbitmq.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.rabbitmq.clientProperties	Connection client properties (client info used in negotiating with the server)	null	false	MEDIUM
camel.component.rabbitmq.connectionFactory ExceptionHandler	Custom rabbitmq ExceptionHandler for ConnectionFactory	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.rabbitmq.connectionTimeout	Connection timeout	60000	false	MEDIUM
camel.component.rabbitmq.networkRecoveryInterval	Network recovery interval in milliseconds (interval used when recovering from network failure)	"5000"	false	MEDIUM
camel.component.rabbitmq.requestedChannelMax	Connection requested channel max (max number of channels offered)	2047	false	MEDIUM
camel.component.rabbitmq.requestedFrameMax	Connection requested frame max (max size of frame offered)	0	false	MEDIUM
camel.component.rabbitmq.requestedHeartbeat	Connection requested heartbeat (heart-beat in seconds offered)	60	false	MEDIUM
camel.component.rabbitmq.requestTimeout	Set timeout for waiting for a reply when using the InOut Exchange Pattern (in milliseconds)	20000L	false	MEDIUM
camel.component.rabbitmq.requestTimeoutChecker Interval	Set requestTimeoutCheckerInterval for inOut exchange	1000L	false	MEDIUM
camel.component.rabbitmq.topologyRecoveryEnabled	Enables connection topology recovery (should topology recovery be performed)	null	false	MEDIUM
camel.component.rabbitmq.transferException	When true and an inOut Exchange failed on the consumer side send the caused Exception back in the response	false	false	MEDIUM
camel.component.rabbitmq.password	Password for authenticated access	"guest"	false	MEDIUM
camel.component.rabbitmq.sslProtocol	Enables SSL on connection, accepted value are true, TLS and 'SSLv3	null	false	MEDIUM
camel.component.rabbitmq.trustManager	Configure SSL trust manager, SSL should be enabled for this option to be effective	null	false	MEDIUM
camel.component.rabbitmq.username	Username in case of authenticated access	"guest"	false	MEDIUM

The camel-rabbitmq source connector has no converters out of the box.

The camel-rabbitmq source connector has no transforms out of the box.

The camel-rabbitmq source connector has no aggregation strategies out of the box.

5.15. SQL

5.15.1. camel-sql-kafka-connector sink configuration

Connector Description: Perform SQL queries using Spring JDBC.

When using camel-sql-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-sql-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.sql.CamelSqlSinkConnector
```

The camel-sql sink connector supports 22 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.query	Sets the SQL query to perform. You can externalize the query by using file: or classpath: as prefix and specify the location of the file.	null	true	HIGH
camel.sink.endpoint.allowNamedParameters	Whether to allow using named parameters in the queries.	true	false	MEDIUM
camel.sink.endpoint.dataSource	Sets the DataSource to use to communicate with the databaset at endpoint level.	null	false	MEDIUM
camel.sink.endpoint.dataSourceRef	Sets the reference to a DataSource to lookup from the registry, to use for communicating with the database.	null	false	LOW
camel.sink.endpoint.outputClass	Specify the full package and class name to use as conversion when outputType=SelectOne.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.outputHeader	Store the query result in a header instead of the message body. By default, outputHeader == null and the query result is stored in the message body, any existing content in the message body is discarded. If outputHeader is set, the value is used as the name of the header to store the query result and the original message body is preserved.	null	false	MEDIUM
camel.sink.endpoint.outputType	Make the output of consumer or producer to SelectList as List of Map, or SelectOne as single Java object in the following way: a) If the query has only single column, then that JDBC Column object is returned. (such as SELECT COUNT( ) FROM PROJECT will return a Long object. b) If the query has more than one column, then it will return a Map of that result. c) If the outputClass is set, then it will convert the query result into an Java bean object by calling all the setters that match the column names. It will assume your class has a default constructor to create instance with. d) If the query resulted in more than one rows, it throws an non-unique result exception. StreamList streams the result of the query using an Iterator. This can be used with the Splitter EIP in streaming mode to process the ResultSet in streaming fashion. One of: [SelectOne] [SelectList] [StreamList]	"SelectList"	false	MEDIUM
camel.sink.endpoint.separator	The separator to use when parameter values is taken from message body (if the body is a String type), to be inserted at # placeholders. Notice if you use named parameters, then a Map type is used instead. The default value is comma	","	false	MEDIUM
camel.sink.endpoint.batch	Enables or disables batch mode	false	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.sink.endpoint.noop	If set, will ignore the results of the SQL query and use the existing IN message as the OUT message for the continuation of processing	false	false	MEDIUM
camel.sink.endpoint.useMessageBodyForSql	Whether to use the message body as the SQL and then headers for parameters. If this option is enabled then the SQL in the uri is not used. Note that query parameters in the message body are represented by a question mark instead of a # symbol.	false	false	MEDIUM
camel.sink.endpoint.alwaysPopulateStatement	If enabled then the populateStatement method from org.apache.camel.component.sql.SqlPrepareStatementStrategy is always invoked, also if there is no expected parameters to be prepared. When this is false then the populateStatement is only invoked if there is 1 or more expected parameters to be set; for example this avoids reading the message body/headers for SQL queries with no parameters.	false	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.parametersCount	If set greater than zero, then Camel will use this count value of parameters to replace instead of querying via JDBC metadata API. This is useful if the JDBC vendor could not return correct parameters count, then user may override instead.	null	false	MEDIUM
camel.sink.endpoint.placeholder	Specifies a character that will be replaced to in SQL query. Notice, that it is simple String.replaceAll() operation and no SQL parsing is involved (quoted strings will also change).	"#"	false	MEDIUM
camel.sink.endpoint.prepareStatementStrategy	Allows to plugin to use a custom org.apache.camel.component.sql.SqlPrepareStatementStrategy to control preparation of the query and prepared statement.	null	false	MEDIUM
camel.sink.endpoint.templateOptions	Configures the Spring JdbcTemplate with the key/values from the Map	null	false	MEDIUM
camel.sink.endpoint.usePlaceholder	Sets whether to use placeholder and replace all placeholder characters with sign in the SQL queries.	true	false	MEDIUM
camel.component.sql.dataSource	Sets the DataSource to use to communicate with the database.	null	false	MEDIUM
camel.component.sql.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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.sql.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.sql.usePlaceholder	Sets whether to use placeholder and replace all placeholder characters with sign in the SQL queries. This option is default true	true	false	MEDIUM

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

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

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

5.15.2. camel-sql-kafka-connector source configuration

Connector description: Perform SQL queries using Spring JDBC.

When using camel-sql-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-sql-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.sql.CamelSqlSourceConnector
```

The camel-sql source connector supports 47 options, which are listed below.

Name	Description	Default	Required	Priority
camel.source.path.query	Sets the SQL query to perform. You can externalize the query by using file: or classpath: as prefix and specify the location of the file.	null	true	HIGH

Name	Description	Default	Required	Priority
camel.source.endpoint.allowNamedParameters	Whether to allow using named parameters in the queries.	true	false	MEDIUM
camel.source.endpoint.dataSource	Sets the DataSource to use to communicate with the databaset at endpoint level.	null	false	MEDIUM
camel.source.endpoint.dataSourceRef	Sets the reference to a DataSource to lookup from the registry, to use for communicating with the database.	null	false	LOW
camel.source.endpoint.outputClass	Specify the full package and class name to use as conversion when outputType=SelectOne.	null	false	MEDIUM
camel.source.endpoint.outputHeader	Store the query result in a header instead of the message body. By default, outputHeader == null and the query result is stored in the message body, any existing content in the message body is discarded. If outputHeader is set, the value is used as the name of the header to store the query result and the original message body is preserved.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.outputType	Make the output of consumer or producer to SelectList as List of Map, or SelectOne as single Java object in the following way: a) If the query has only single column, then that JDBC Column object is returned. (such as SELECT COUNT( ) FROM PROJECT will return a Long object. b) If the query has more than one column, then it will return a Map of that result. c) If the outputClass is set, then it will convert the query result into an Java bean object by calling all the setters that match the column names. It will assume your class has a default constructor to create instance with. d) If the query resulted in more than one rows, it throws an non-unique result exception. StreamList streams the result of the query using an Iterator. This can be used with the Splitter EIP in streaming mode to process the ResultSet in streaming fashion. One of: [SelectOne] [SelectList] [StreamList]	"SelectList"	false	MEDIUM
camel.source.endpoint.separator	The separator to use when parameter values is taken from message body (if the body is a String type), to be inserted at # placeholders. Notice if you use named parameters, then a Map type is used instead. The default value is comma	","	false	MEDIUM
camel.source.endpoint.breakBatchOnConsumeFail	Sets whether to break batch if onConsume failed.	false	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.source.endpoint.expectedUpdateCount	Sets an expected update count to validate when using onConsume.	-1	false	MEDIUM
camel.source.endpoint.maxMessagesPerPoll	Sets the maximum number of messages to poll	null	false	MEDIUM
camel.source.endpoint.onConsume	After processing each row then this query can be executed, if the Exchange was processed successfully, for example to mark the row as processed. The query can have parameter.	null	false	MEDIUM
camel.source.endpoint.onConsumeBatchComplete	After processing the entire batch, this query can be executed to bulk update rows etc. The query cannot have parameters.	null	false	MEDIUM
camel.source.endpoint.onConsumeFailed	After processing each row then this query can be executed, if the Exchange failed, for example to mark the row as failed. The query can have parameter.	null	false	MEDIUM
camel.source.endpoint.routeEmptyResultSet	Sets whether empty resultset should be allowed to be sent to the next hop. Defaults to false. So the empty resultset will be filtered out.	false	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.transacted	Enables or disables transaction. If enabled then if processing an exchange failed then the consumer breaks out processing any further exchanges to cause a rollback eager.	false	false	MEDIUM
camel.source.endpoint.useIterator	Sets how resultset should be delivered to route. Indicates delivery as either a list or individual object. defaults to true.	true	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	false	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	false	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	false	MEDIUM
camel.source.endpoint.processingStrategy	Allows to plugin to use a custom org.apache.camel.component.sql.SqlProcessingStrategy to execute queries when the consumer has processed the rows/batch.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.alwaysPopulateStatement	If enabled then the populateStatement method from org.apache.camel.component.sql.SqlPrepareStatementStrategy is always invoked, also if there is no expected parameters to be prepared. When this is false then the populateStatement is only invoked if there is 1 or more expected parameters to be set; for example this avoids reading the message body/headers for SQL queries with no parameters.	false	false	MEDIUM
camel.source.endpoint.parametersCount	If set greater than zero, then Camel will use this count value of parameters to replace instead of querying via JDBC metadata API. This is useful if the JDBC vendor could not return correct parameters count, then user may override instead.	null	false	MEDIUM
camel.source.endpoint.placeholder	Specifies a character that will be replaced to in SQL query. Notice, that it is simple String.replaceAll() operation and no SQL parsing is involved (quoted strings will also change).	"#"	false	MEDIUM
camel.source.endpoint.prepareStatementStrategy	Allows to plugin to use a custom org.apache.camel.component.sql.SqlPrepareStatementStrategy to control preparation of the query and prepared statement.	null	false	MEDIUM
camel.source.endpoint.templateOptions	Configures the Spring JdbcTemplate with the key/values from the Map	null	false	MEDIUM
camel.source.endpoint.usePlaceholder	Sets whether to use placeholder and replace all placeholder characters with sign in the SQL queries.	true	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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.backoffIdleThreshold	The number of subsequent idle polls that should happen before the backoffMultipler should kick-in.	null	false	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	false	MEDIUM
camel.source.endpoint.delay	Milliseconds before the next poll.	500L	false	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	false	MEDIUM
camel.source.endpoint.initialDelay	Milliseconds before the first poll starts.	1000L	false	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	false	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"	false	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	false	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"	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.schedulerProperties	To configure additional properties when using a custom scheduler or any of the Quartz, Spring based scheduler.	null	false	MEDIUM
camel.source.endpoint.startScheduler	Whether the scheduler should be auto started.	true	false	MEDIUM
camel.source.endpoint.timeUnit	Time unit for initialDelay and delay options. One of: [NANOSECONDS] [MICROSECONDS] [MILLISECONDS] [SECONDS] [MINUTES] [HOURS] [DAYS]	"MILLISECONDS"	false	MEDIUM
camel.source.endpoint.useFixedDelay	Controls if fixed delay or fixed rate is used. See ScheduledExecutorService in JDK for details.	true	false	MEDIUM
camel.component.sql.dataSource	Sets the DataSource to use to communicate with the database.	null	false	MEDIUM
camel.component.sql.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	false	MEDIUM
camel.component.sql.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.sql.usePlaceholder	Sets whether to use placeholder and replace all placeholder characters with sign in the SQL queries. This option is default true	true	false	MEDIUM

The camel-sql source connector has no converters out of the box.

The camel-sql source connector has no transforms out of the box.

The camel-sql source connector has no aggregation strategies out of the box.

5.16. SSH

5.16.1. camel-ssh-kafka-connector sink configuration

Connector Description: Execute commands on remote hosts using SSH.

When using camel-ssh-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-ssh-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.ssh.CamelSshSinkConnector
```

The camel-ssh sink connector supports 30 options, which are listed below.

Name	Description	Default	Required	Priority
camel.sink.path.host	Sets the hostname of the remote SSH server.	null	true	HIGH
camel.sink.path.port	Sets the port number for the remote SSH server.	22	false	MEDIUM
camel.sink.endpoint.failOnUnknownHost	Specifies whether a connection to an unknown host should fail or not. This value is only checked when the property knownHosts is set.	false	false	MEDIUM
camel.sink.endpoint.knownHostsResource	Sets the resource path for a known_hosts file	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.timeout	Sets the timeout in milliseconds to wait in establishing the remote SSH server connection. Defaults to 30000 milliseconds.	30000L	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	false	MEDIUM
camel.sink.endpoint.channelType	Sets the channel type to pass to the Channel as part of command execution. Defaults to exec.	"exec"	false	MEDIUM
camel.sink.endpoint.shellPrompt	Sets the shellPrompt to be dropped when response is read after command execution	null	false	MEDIUM
camel.sink.endpoint.sleepForShellPrompt	Sets the sleep period in milliseconds to wait reading response from shell prompt. Defaults to 100 milliseconds.	100L	false	MEDIUM
camel.sink.endpoint.certResource	Sets the resource path of the certificate to use for Authentication. Will use ResourceHelperKeyPairProvider to resolve file based certificate, and depends on keyType setting.	null	false	MEDIUM
camel.sink.endpoint.certResourcePassword	Sets the password to use in loading certResource, if certResource is an encrypted key.	null	false	MEDIUM
camel.sink.endpoint.keyPairProvider	Sets the KeyPairProvider reference to use when connecting using Certificates to the remote SSH Server.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.sink.endpoint.keyType	Sets the key type to pass to the KeyPairProvider as part of authentication. KeyPairProvider.loadKey(...) will be passed this value. From Camel 3.0.0 / 2.25.0, by default Camel will select the first available KeyPair that is loaded. Prior to this, a KeyType of 'ssh-rsa' was enforced by default.	null	false	MEDIUM
camel.sink.endpoint.password	Sets the password to use in connecting to remote SSH server. Requires keyPairProvider to be set to null.	null	false	MEDIUM
camel.sink.endpoint.username	Sets the username to use in logging into the remote SSH server.	null	false	MEDIUM
camel.component.ssh.failOnUnknownHost	Specifies whether a connection to an unknown host should fail or not. This value is only checked when the property knownHosts is set.	false	false	MEDIUM
camel.component.ssh.knownHostsResource	Sets the resource path for a known_hosts file	null	false	MEDIUM
camel.component.ssh.timeout	Sets the timeout in milliseconds to wait in establishing the remote SSH server connection. Defaults to 30000 milliseconds.	30000L	false	MEDIUM
camel.component.ssh.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	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.ssh.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM
camel.component.ssh.channelType	Sets the channel type to pass to the Channel as part of command execution. Defaults to exec.	"exec"	false	MEDIUM
camel.component.ssh.configuration	Component configuration	null	false	MEDIUM
camel.component.ssh.shellPrompt	Sets the shellPrompt to be dropped when response is read after command execution	null	false	MEDIUM
camel.component.ssh.sleepForShellPrompt	Sets the sleep period in milliseconds to wait reading response from shell prompt. Defaults to 100 milliseconds.	100L	false	MEDIUM
camel.component.ssh.certResource	Sets the resource path of the certificate to use for Authentication. Will use ResourceHelperKeyPairProvider to resolve file based certificate, and depends on keyType setting.	null	false	MEDIUM
camel.component.ssh.certResourcePassword	Sets the password to use in loading certResource, if certResource is an encrypted key.	null	false	MEDIUM
camel.component.ssh.keyPairProvider	Sets the KeyPairProvider reference to use when connecting using Certificates to the remote SSH Server.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.ssh.keyType	Sets the key type to pass to the KeyPairProvider as part of authentication. KeyPairProvider.loadKey(…) will be passed this value. From Camel 3.0.0 / 2.25.0, by default Camel will select the first available KeyPair that is loaded. Prior to this, a KeyType of 'ssh-rsa' was enforced by default.	null	false	MEDIUM
camel.component.ssh.password	Sets the password to use in connecting to remote SSH server. Requires keyPairProvider to be set to null.	null	false	MEDIUM
camel.component.ssh.username	Sets the username to use in logging into the remote SSH server.	null	false	MEDIUM

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

The camel-ssh sink connector supports 0 transforms out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.ssh.transformers.SshTransforms
```

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

5.16.2. camel-ssh-kafka-connector source configuration

Connector description: Execute commands on remote hosts using SSH.

When using camel-ssh-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-ssh-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.ssh.CamelSshSourceConnector
```

The camel-ssh source connector supports 50 options, which are listed below.

Name	Description	Default	Required	Priority
camel.source.path.host	Sets the hostname of the remote SSH server.	null	true	HIGH
camel.source.path.port	Sets the port number for the remote SSH server.	22	false	MEDIUM
camel.source.endpoint.failOnUnknownHost	Specifies whether a connection to an unknown host should fail or not. This value is only checked when the property knownHosts is set.	false	false	MEDIUM
camel.source.endpoint.knownHostsResource	Sets the resource path for a known_hosts file	null	false	MEDIUM
camel.source.endpoint.timeout	Sets the timeout in milliseconds to wait in establishing the remote SSH server connection. Defaults to 30000 milliseconds.	30000L	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	false	MEDIUM
camel.source.endpoint.pollCommand	Sets the command string to send to the remote SSH server during every poll cycle. Only works with camel-ssh component being used as a consumer, i.e. from(ssh://...) You may need to end your command with a newline, and that must be URL encoded %0A	null	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	false	MEDIUM

Name	Description	Default	Required	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	false	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	false	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	false	MEDIUM
camel.source.endpoint.channelType	Sets the channel type to pass to the Channel as part of command execution. Defaults to exec.	"exec"	false	MEDIUM
camel.source.endpoint.shellPrompt	Sets the shellPrompt to be dropped when response is read after command execution	null	false	MEDIUM
camel.source.endpoint.sleepForShellPrompt	Sets the sleep period in milliseconds to wait reading response from shell prompt. Defaults to 100 milliseconds.	100L	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	false	MEDIUM
camel.source.endpoint.backoffIdleThreshold	The number of subsequent idle polls that should happen before the backoffMultipler should kick-in.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.backoffMultiplier	To let the scheduled polling consumer backoff if there has been a number of subsequent idles/errors in a row. The multiplier is then the number of polls that will be skipped before the next actual attempt is happening again. When this option is in use then backoffIdleThreshold and/or backoffErrorThreshold must also be configured.	null	false	MEDIUM
camel.source.endpoint.delay	Milliseconds before the next poll.	500L	false	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	false	MEDIUM
camel.source.endpoint.initialDelay	Milliseconds before the first poll starts.	1000L	false	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	false	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"	false	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	false	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"	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.schedulerProperties	To configure additional properties when using a custom scheduler or any of the Quartz, Spring based scheduler.	null	false	MEDIUM
camel.source.endpoint.startScheduler	Whether the scheduler should be auto started.	true	false	MEDIUM
camel.source.endpoint.timeUnit	Time unit for initialDelay and delay options. One of: [NANOSECONDS] [MICROSECONDS] [MILLISECONDS] [SECONDS] [MINUTES] [HOURS] [DAYS]	"MILLISECONDS"	false	MEDIUM
camel.source.endpoint.useFixedDelay	Controls if fixed delay or fixed rate is used. See ScheduledExecutorService in JDK for details.	true	false	MEDIUM
camel.source.endpoint.certResource	Sets the resource path of the certificate to use for Authentication. Will use ResourceHelperKeyPairProvider to resolve file based certificate, and depends on keyType setting.	null	false	MEDIUM
camel.source.endpoint.certResourcePassword	Sets the password to use in loading certResource, if certResource is an encrypted key.	null	false	MEDIUM
camel.source.endpoint.keyPairProvider	Sets the KeyPairProvider reference to use when connecting using Certificates to the remote SSH Server.	null	false	MEDIUM
camel.source.endpoint.keyType	Sets the key type to pass to the KeyPairProvider as part of authentication. KeyPairProvider.loadKey(...) will be passed this value. From Camel 3.0.0 / 2.25.0, by default Camel will select the first available KeyPair that is loaded. Prior to this, a KeyType of 'ssh-rsa' was enforced by default.	null	false	MEDIUM
camel.source.endpoint.password	Sets the password to use in connecting to remote SSH server. Requires keyPairProvider to be set to null.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.source.endpoint.username	Sets the username to use in logging into the remote SSH server.	null	false	MEDIUM
camel.component.ssh.failOnUnknownHost	Specifies whether a connection to an unknown host should fail or not. This value is only checked when the property knownHosts is set.	false	false	MEDIUM
camel.component.ssh.knownHostsResource	Sets the resource path for a known_hosts file	null	false	MEDIUM
camel.component.ssh.timeout	Sets the timeout in milliseconds to wait in establishing the remote SSH server connection. Defaults to 30000 milliseconds.	30000L	false	MEDIUM
camel.component.ssh.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	false	MEDIUM
camel.component.ssh.pollCommand	Sets the command string to send to the remote SSH server during every poll cycle. Only works with camel-ssh component being used as a consumer, i.e. from(ssh://...) You may need to end your command with a newline, and that must be URL encoded %0A	null	false	MEDIUM
camel.component.ssh.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.ssh.channelType	Sets the channel type to pass to the Channel as part of command execution. Defaults to exec.	"exec"	false	MEDIUM
camel.component.ssh.configuration	Component configuration	null	false	MEDIUM
camel.component.ssh.shellPrompt	Sets the shellPrompt to be dropped when response is read after command execution	null	false	MEDIUM
camel.component.ssh.sleepForShellPrompt	Sets the sleep period in milliseconds to wait reading response from shell prompt. Defaults to 100 milliseconds.	100L	false	MEDIUM
camel.component.ssh.certResource	Sets the resource path of the certificate to use for Authentication. Will use ResourceHelperKeyPairProvider to resolve file based certificate, and depends on keyType setting.	null	false	MEDIUM
camel.component.ssh.certResourcePassword	Sets the password to use in loading certResource, if certResource is an encrypted key.	null	false	MEDIUM
camel.component.ssh.keyPairProvider	Sets the KeyPairProvider reference to use when connecting using Certificates to the remote SSH Server.	null	false	MEDIUM
camel.component.ssh.keyType	Sets the key type to pass to the KeyPairProvider as part of authentication. KeyPairProvider.loadKey(...) will be passed this value. From Camel 3.0.0 / 2.25.0, by default Camel will select the first available KeyPair that is loaded. Prior to this, a KeyType of 'ssh-rsa' was enforced by default.	null	false	MEDIUM
camel.component.ssh.password	Sets the password to use in connecting to remote SSH server. Requires keyPairProvider to be set to null.	null	false	MEDIUM

Name	Description	Default	Required	Priority
camel.component.ssh.username	Sets the username to use in logging into the remote SSH server.	null	false	MEDIUM

The camel-ssh source connector has no converters out of the box.

The camel-ssh source connector supports 0 transforms out of the box, which are listed below.

```
org.apache.camel.kafkaconnector.ssh.transformers.SshTransforms
```

The camel-ssh source connector has no aggregation strategies out of the box.

5.17. SYSLOG

5.17.1. camel-syslog-kafka-connector sink configuration

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

The camel-syslog sink connector supports is based on camel-netty sink 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.sink.path.protocol	The protocol to use which can be tcp or udp. One of: [tcp] [udp]	null	HIGH
camel.sink.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.sink.path.port	The host port number	null	HIGH

5.17.2. 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

5.18. TIMER

5.18.1. camel-timer-kafka-connector source configuration

Connector description: Generate messages in specified intervals using java.util.Timer.

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 16 options, which are listed below.

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

Name	Description	Default	Required	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	false	MEDIUM
camel.source.endpoint.delay	Delay before first event is triggered.	1000L	false	MEDIUM
camel.source.endpoint.fixedRate	Events take place at approximately regular intervals, separated by the specified period.	false	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	false	MEDIUM
camel.source.endpoint.period	If greater than 0, generate periodic events every period.	1000L	false	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	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	false	MEDIUM
camel.source.endpoint.exchangePattern	Sets the exchange pattern when the consumer creates an exchange. One of: [InOnly] [InOut] [InOptionalOut]	null	false	MEDIUM

Name	Description	Default	Required	Priority
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	false	MEDIUM
camel.source.endpoint.pattern	Allows you to specify a custom Date pattern to use for setting the time option using URI syntax.	null	false	MEDIUM
camel.source.endpoint.synchronous	Sets whether synchronous processing should be strictly used	false	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	false	MEDIUM
camel.source.endpoint.timer	To use a custom Timer	null	false	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	false	MEDIUM
camel.component.timer.autowiredEnabled	Whether autowiring is enabled. This is used for automatic autowiring options (the option must be marked as autowired) by looking up in the registry to find if there is a single instance of matching type, which then gets configured on the component. This can be used for automatic configuring JDBC data sources, JMS connection factories, AWS Clients, etc.	true	false	MEDIUM

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

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

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