Red Hat Integration 2022.Q2

Kamelets Reference

Kamelets Reference

Last Updated: 2022-06-16

Red Hat Integration 2022.Q2 Kamelets Reference

Kamelets Reference

Legal Notice

Abstract

Kamelets are reusable route components that hide the complexity of creating data pipelines that connect to external systems.

Table of Contents

PREFACE

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. AVRO DESERIALIZE ACTION

Deserialize payload to Avro

1.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **avro-deserialize-action** Kamelet:

Property	Name	Description	Type	Default	Example
schema *	Schema	The Avro schema to use during serialization (as single-line, using JSON format)	string		**"{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\": [{\"name\": \"first\", \"type\": \"string\"}, {\"name\": \"last\", \"type\": \"string\"}]}"**
validate	Validate	Indicates if the content must be validated against the schema	boolean	**true**	

NOTE

Fields marked with an asterisk (*) are mandatory.

1.2. DEPENDENCIES

At runtime, the **avro-deserialize-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson-avro

1.3. USAGE

This section describes how you can use the **avro-deserialize-action**.

1.3.1. Knative Action

You can use the **avro-deserialize-action** Kamelet as an intermediate step in a Knative binding.

avro-deserialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: avro-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first":"Ada","last":"Lovelace"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-serialize-action
    properties:
      schema: "{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\":
[{\"name\": \"first\", \"type\": \"string\"},{\"name\": \"last\", \"type\": \"string\"}]}"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-deserialize-action
    properties:
      schema: "{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\":
[{\"name\": \"first\", \"type\": \"string\"},{\"name\": \"last\", \"type\": \"string\"}]}"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

1.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

1.3.1.2. Procedure for using the cluster CLI

1. Open the **avro-deserialize-action-binding.yaml** file in an editor, and then edit it

1. Save the **avro-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f avro-deserialize-action-binding.yaml
```

1.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name avro-deserialize-action-binding timer-source?
message='{"first":"Ada","last":"Lovelace"}' --step json-deserialize-action --step avro-serialize-action -p
step-1.schema='{"type": "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"},{"name": "last", "type": "string"}]}' --step avro-deserialize-action -p
step-2.schema='{"type": "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"},{"name": "last", "type": "string"}]}' --step json-serialize-action
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

1.3.2. Kafka Action

You can use the **avro-deserialize-action** Kamelet as an intermediate step in a Kafka binding.

avro-deserialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: avro-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first":"Ada","last":"Lovelace"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-serialize-action
    properties:
      schema: "{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\":
[{\"name\": \"first\", \"type\": \"string\"},{\"name\": \"last\", \"type\": \"string\"}]}"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-deserialize-action
```

```
    properties:
      schema: "{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\":
[{\"name\": \"first\", \"type\": \"string\"},{\"name\": \"last\", \"type\": \"string\"}]}"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

1.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

1.3.2.2. Procedure for using the cluster CLI

1. Save the **avro-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f avro-deserialize-action-binding.yaml
```

1.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name avro-deserialize-action-binding timer-source?
message='{"first":"Ada","last":"Lovelace"}' --step json-deserialize-action --step avro-serialize-action -p
step-1.schema='{"type": "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"},{"name": "last", "type": "string"}]}' --step avro-deserialize-action -p
step-2.schema='{"type": "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"},{"name": "last", "type": "string"}]}' --step json-serialize-action
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

1.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/avro-deserialize-action.kamelet.yaml

CHAPTER 2. AVRO SERIALIZE ACTION

Serialize payload to Avro

2.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **avro-serialize-action** Kamelet:

Property	Name	Description	Type	Default	Example
schema *	Schema	The Avro schema to use during serialization (as single-line, using JSON format)	string		**"{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\": [{\"name\": \"first\", \"type\": \"string\"}, {\"name\": \"last\", \"type\": \"string\"}]}"**
validate	Validate	Indicates if the content must be validated against the schema	boolean	**true**	

NOTE

Fields marked with an asterisk (*) are mandatory.

2.2. DEPENDENCIES

At runtime, the **avro-serialize-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson-avro

2.3. USAGE

This section describes how you can use the **avro-serialize-action**.

2.3.1. Knative Action

You can use the **avro-serialize-action** Kamelet as an intermediate step in a Knative binding.

avro-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: avro-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first":"Ada","last":"Lovelace"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-serialize-action
    properties:
      schema: "{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\": [{\"name\": \"first\", \"type\": \"string\"},{\"name\": \"last\", \"type\": \"string\"}]}"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

2.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

2.3.1.2. Procedure for using the cluster CLI

1. Save the **avro-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f avro-serialize-action-binding.yaml
```

2.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name avro-serialize-action-binding timer-source?
message='{"first":"Ada","last":"Lovelace"}' --step json-deserialize-action --step avro-serialize-action -p
step-1.schema='{"type": "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"},{"name": "last", "type": "string"}]}' channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

2.3.2. Kafka Action

You can use the **avro-serialize-action** Kamelet as an intermediate step in a Kafka binding.

avro-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: avro-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first":"Ada","last":"Lovelace"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: avro-serialize-action
    properties:
      schema: "{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\":
[{\"name\": \"first\", \"type\": \"string\"},{\"name\": \"last\", \"type\": \"string\"}]}"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

2.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

2.3.2.2. Procedure for using the cluster CLI

1. Save the **avro-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f avro-serialize-action-binding.yaml
```

2.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name avro-serialize-action-binding timer-source?
message='{"first":"Ada","last":"Lovelace"}' --step json-deserialize-action --step avro-serialize-action -p
step-1.schema='{"type": "record", "namespace": "com.example", "name": "FullName", "fields":
[{"name": "first", "type": "string"},{"name": "last", "type": "string"}]}'
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

2.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/avro-serialize-action.kamelet.yaml

CHAPTER 3. AWS KINESIS SINK

Send data to AWS Kinesis.

The Kamelet expects the following header:

- **partition** / **ce-partition**: to set the Kinesis partition key

If the header won't be set the exchange ID will be used.

The Kamelet is also able to recognize the following header:

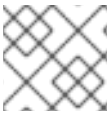
- **sequence-number** / **ce-sequencenumber**: to set the Sequence number

This header is optional.

3.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-kinesis-sink** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
stream *	Stream Name	The Kinesis stream that you want to access (needs to be created in advance)	string		



NOTE

Fields marked with an asterisk (*) are mandatory.

3.2. DEPENDENCIES

At runtime, the **aws-kinesis-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-kinesis
- camel:kamelet

3.3. USAGE

This section describes how you can use the **aws-kinesis-sink**.

3.3.1. Knative Sink

You can use the **aws-kinesis-sink** Kamelet as a Knative sink by binding it to a Knative object.

aws-kinesis-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-kinesis-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-kinesis-sink
    properties:
      accessKey: "The Access Key"
      region: "eu-west-1"
      secretKey: "The Secret Key"
      stream: "The Stream Name"
```

3.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

3.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-kinesis-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-kinesis-sink-binding.yaml
```

3.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-kinesis-sink -p "sink.accessKey=The Access Key" -p
"sink.region=eu-west-1" -p "sink.secretKey=The Secret Key" -p "sink.stream=The Stream Name"
```

This command creates the KameletBinding in the current namespace on the cluster.

3.3.2. Kafka Sink

You can use the **aws-kinesis-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

aws-kinesis-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-kinesis-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-kinesis-sink
    properties:
      accessKey: "The Access Key"
      region: "eu-west-1"
      secretKey: "The Secret Key"
      stream: "The Stream Name"
```

3.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

3.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-kinesis-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-kinesis-sink-binding.yaml
```

3.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-kinesis-sink -p "sink.accessKey=The Access Key" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key" -p "sink.stream=The Stream Name"
```

This command creates the KameletBinding in the current namespace on the cluster.

3.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-kinesis-sink.kamelet.yaml

CHAPTER 4. AWS KINESIS SOURCE

Receive data from AWS Kinesis.

4.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-kinesis-source** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
stream *	Stream Name	The Kinesis stream that you want to access (needs to be created in advance)	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

4.2. DEPENDENCIES

At runtime, the **aws-kinesis-source** Kamelet relies upon the presence of the following dependencies:

- camel:gson
- camel:kamelet
- camel:aws2-kinesis

4.3. USAGE

This section describes how you can use the **aws-kinesis-source**.

4.3.1. Knative Source

You can use the **aws-kinesis-source** Kamelet as a Knative source by binding it to a Knative object.

aws-kinesis-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
```

```
kind: KameletBinding
metadata:
  name: aws-kinesis-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-kinesis-source
    properties:
      accessKey: "The Access Key"
      region: "eu-west-1"
      secretKey: "The Secret Key"
      stream: "The Stream Name"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

4.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

4.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-kinesis-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f aws-kinesis-source-binding.yaml
```

4.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-kinesis-source -p "source.accessKey=The Access Key" -p "source.region=eu-west-1"
-p "source.secretKey=The Secret Key" -p "source.stream=The Stream Name" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

4.3.2. Kafka Source

You can use the **aws-kinesis-source** Kamelet as a Kafka source by binding it to a Kafka topic.

aws-kinesis-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-kinesis-source-binding
```

```
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-kinesis-source
    properties:
      accessKey: "The Access Key"
      region: "eu-west-1"
      secretKey: "The Secret Key"
      stream: "The Stream Name"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

4.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

4.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-kinesis-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f aws-kinesis-source-binding.yaml
```

4.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-kinesis-source -p "source.accessKey=The Access Key" -p "source.region=eu-west-1"
-p "source.secretKey=The Secret Key" -p "source.stream=The Stream Name"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

4.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-kinesis-source.kamelet.yaml

CHAPTER 5. AWS LAMBDA SINK

Send a payload to an AWS Lambda function

5.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-lambda-sink** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS	string		
function *	Function Name	The Lambda Function name	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

5.2. DEPENDENCIES

At runtime, the **aws-lambda-sink** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:aws2-lambda

5.3. USAGE

This section describes how you can use the **aws-lambda-sink**.

5.3.1. Knative Sink

You can use the **aws-lambda-sink** Kamelet as a Knative sink by binding it to a Knative object.

aws-lambda-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-lambda-sink-binding
spec:
  source:
```

```
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-lambda-sink
    properties:
      accessKey: "The Access Key"
      function: "The Function Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
```

5.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

5.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-lambda-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-lambda-sink-binding.yaml
```

5.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-lambda-sink -p "sink.accessKey=The Access Key" -p
"sink.function=The Function Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

5.3.2. Kafka Sink

You can use the **aws-lambda-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

aws-lambda-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-lambda-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
```

```
    name: my-topic
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: aws-lambda-sink
  properties:
    accessKey: "The Access Key"
    function: "The Function Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"
```

5.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

5.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-lambda-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-lambda-sink-binding.yaml
```

5.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-lambda-sink -p "sink.accessKey=The Access Key" -p "sink.function=The Function Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

5.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-lambda-sink.kamelet.yaml

CHAPTER 6. AWS REDSHIFT SINK

Send data to an AWS Redshift Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'

The Kamelet needs to receive as input something like:

'{ "username":"oscerd", "city":"Rome"}'

6.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-redshift-sink** Kamelet:

Property	Name	Description	Type	Default	Example
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured AWS Redshift Database	string		
query *	Query	The Query to execute against the AWS Redshift Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured AWS Redshift Database	string		
serverPort	Server Port	Server Port for the data source	string	**5439**	

NOTE

Fields marked with an asterisk (*) are mandatory.

6.2. DEPENDENCIES

At runtime, the **aws-redshift-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:com.amazon.redshift:redshift-jdbc42:2.1.0.5
- mvn:org.apache.commons:commons-dbcp2:2.7.0

6.3. USAGE

This section describes how you can use the **aws-redshift-sink**.

6.3.1. Knative Sink

You can use the **aws-redshift-sink** Kamelet as a Knative sink by binding it to a Knative object.

aws-redshift-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-redshift-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-redshift-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

6.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

6.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-redshift-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-redshift-sink-binding.yaml
```

6.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-redshift-sink -p "sink.databaseName=The Database Name" -p
"sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES
(:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

6.3.2. Kafka Sink

You can use the **aws-redshift-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

aws-redshift-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-redshift-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-redshift-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

6.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

6.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-redshift-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-redshift-sink-binding.yaml
```

6.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-redshift-sink -p
"sink.databaseName=The Database Name" -p "sink.password=The Password" -p
"sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p
"sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

6.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-redshift-sink.kamelet.yaml

CHAPTER 7. AWS SNS SINK

Send message to an AWS SNS Topic

7.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-sns-sink** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
topicNameOrArn *	Topic Name	The SQS Topic name or ARN	string		
autoCreateTopic	Autocreate Topic	Setting the autocreation of the SNS topic.	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

7.2. DEPENDENCIES

At runtime, the **aws-sns-sink** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:aws2-sns

7.3. USAGE

This section describes how you can use the **aws-sns-sink**.

7.3.1. Knative Sink

You can use the **aws-sns-sink** Kamelet as a Knative sink by binding it to a Knative object.

aws-sns-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
```

```
metadata:
  name: aws-sns-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sns-sink
    properties:
      accessKey: "The Access Key"
      region: "eu-west-1"
      secretKey: "The Secret Key"
      topicNameOrArn: "The Topic Name"
```

7.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

7.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-sns-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-sns-sink-binding.yaml
```

7.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-sns-sink -p "sink.accessKey=The Access Key" -p
"sink.region=eu-west-1" -p "sink.secretKey=The Secret Key" -p "sink.topicNameOrArn=The Topic
Name"
```

This command creates the KameletBinding in the current namespace on the cluster.

7.3.2. Kafka Sink

You can use the **aws-sns-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

aws-sns-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sns-sink-binding
```

```
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sns-sink
    properties:
      accessKey: "The Access Key"
      region: "eu-west-1"
      secretKey: "The Secret Key"
      topicNameOrArn: "The Topic Name"
```

7.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

7.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-sns-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-sns-sink-binding.yaml
```

7.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-sns-sink -p "sink.accessKey=The Access Key" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key" -p "sink.topicNameOrArn=The Topic Name"
```

This command creates the KameletBinding in the current namespace on the cluster.

7.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-sns-sink.kamelet.yaml

CHAPTER 8. AWS SQS SINK

Send message to an AWS SQS Queue

8.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-sqs-sink** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS	string		
queueNameOrArn *	Queue Name	The SQS Queue name or ARN	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
autoCreateQueue	Autocreate Queue	Setting the autocreation of the SQS queue.	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

8.2. DEPENDENCIES

At runtime, the **aws-sqs-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-sqs
- camel:core
- camel:kamelet

8.3. USAGE

This section describes how you can use the **aws-sqs-sink**.

8.3.1. Knative Sink

You can use the **aws-sqs-sink** Kamelet as a Knative sink by binding it to a Knative object.

aws-sqs-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-sink
    properties:
      accessKey: "The Access Key"
      queueNameOrArn: "The Queue Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
```

8.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

8.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-sqs-sink-binding.yaml
```

8.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-sqs-sink -p "sink.accessKey=The Access Key" -p
"sink.queueNameOrArn=The Queue Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The
Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

8.3.2. Kafka Sink

You can use the **aws-sqs-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

aws-sqs-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
```

```
metadata:
  name: aws-sqs-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-sink
    properties:
      accessKey: "The Access Key"
      queueNameOrArn: "The Queue Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
```

8.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

8.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-sqs-sink-binding.yaml
```

8.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-sqs-sink -p "sink.accessKey=The Access Key" -p "sink.queueNameOrArn=The Queue Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

8.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-sqs-sink.kamelet.yaml

CHAPTER 9. AWS SQS SOURCE

Receive data from AWS SQS.

9.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-sqs-source** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS	string		
queueNameOrArn *	Queue Name	The SQS Queue name or ARN	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
autoCreateQueue	Autocreate Queue	Setting the autocreation of the SQS queue.	boolean	**false**	
deleteAfterRead	Auto-delete Messages	Delete messages after consuming them	boolean	**true**	

NOTE

Fields marked with an asterisk (*) are mandatory.

9.2. DEPENDENCIES

At runtime, the **aws-sqs-source** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-sqs
- camel:core
- camel:kamelet
- camel:jackson

9.3. USAGE

This section describes how you can use the **aws-sqs-source**.

9.3.1. Knative Source

You can use the **aws-sqs-source** Kamelet as a Knative source by binding it to a Knative object.

aws-sqs-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-source
    properties:
      accessKey: "The Access Key"
      queueNameOrArn: "The Queue Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

9.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

9.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f aws-sqs-source-binding.yaml
```

9.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-sqs-source -p "source.accessKey=The Access Key" -p
"source.queueNameOrArn=The Queue Name" -p "source.region=eu-west-1" -p
"source.secretKey=The Secret Key" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

9.3.2. Kafka Source

You can use the **aws-sqs-source** Kamelet as a Kafka source by binding it to a Kafka topic.

aws-sqs-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-source
    properties:
      accessKey: "The Access Key"
      queueNameOrArn: "The Queue Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

9.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

9.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f aws-sqs-source-binding.yaml
```

9.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-sqs-source -p "source.accessKey=The Access Key" -p
"source.queueNameOrArn=The Queue Name" -p "source.region=eu-west-1" -p
"source.secretKey=The Secret Key" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

9.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-sqs-source.kamelet.yaml

CHAPTER 10. AWS 2 SIMPLE QUEUE SERVICE FIFO SINK

Send message to an AWS SQS FIFO Queue

10.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-sqs-fifo-sink** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS	string		
queueNameOrArn *	Queue Name	The SQS Queue name or ARN	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
autoCreateQueue	Autocreate Queue	Setting the autocreation of the SQS queue.	boolean	**false**	
contentBasedDeduplication	Content-Based Deduplication	Use content-based deduplication (should be enabled in the SQS FIFO queue first)	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

10.2. DEPENDENCIES

At runtime, the **aws-sqs-fifo-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-sqs
- camel:core
- camel:kamelet

10.3. USAGE

This section describes how you can use the **aws-sqs-fifo-sink**.

10.3.1. Knative Sink

You can use the **aws-sqs-fifo-sink** Kamelet as a Knative sink by binding it to a Knative object.

aws-sqs-fifo-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-fifo-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-fifo-sink
    properties:
      accessKey: "The Access Key"
      queueNameOrArn: "The Queue Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
```

10.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

10.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-fifo-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-sqs-fifo-sink-binding.yaml
```

10.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-sqs-fifo-sink -p "sink.accessKey=The Access Key" -p
"sink.queueNameOrArn=The Queue Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The
Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

10.3.2. Kafka Sink

You can use the **aws-sqs-fifo-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

aws-sqs-fifo-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-sqs-fifo-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-sqs-fifo-sink
    properties:
      accessKey: "The Access Key"
      queueNameOrArn: "The Queue Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
```

10.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

10.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-sqs-fifo-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-sqs-fifo-sink-binding.yaml
```

10.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-sqs-fifo-sink -p "sink.accessKey=The Access Key" -p "sink.queueNameOrArn=The Queue Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

10.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-sqs-fifo-sink.kamelet.yaml

CHAPTER 11. AWS S3 SINK

Upload data to AWS S3.

The Kamelet expects the following headers to be set:

- **file** / **ce-file**: as the file name to upload

If the header won't be set the exchange ID will be used as file name.

11.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-s3-sink** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS.	string		
bucketNameOrArn *	Bucket Name	The S3 Bucket name or ARN.	string		
region *	AWS Region	The AWS region to connect to.	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS.	string		
autoCreateBucket	Autocreate Bucket	Setting the autocreation of the S3 bucket bucketName.	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

11.2. DEPENDENCIES

At runtime, the **aws-s3-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-s3
- camel:kamelet

11.3. USAGE

This section describes how you can use the **aws-s3-sink**.

11.3.1. Knative Sink

You can use the **aws-s3-sink** Kamelet as a Knative sink by binding it to a Knative object.

aws-s3-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-s3-sink
    properties:
      accessKey: "The Access Key"
      bucketNameOrArn: "The Bucket Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
```

11.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

11.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-s3-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f aws-s3-sink-binding.yaml
```

11.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-s3-sink -p "sink.accessKey=The Access Key" -p
"sink.bucketNameOrArn=The Bucket Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The
Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

11.3.2. Kafka Sink

You can use the **aws-s3-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

aws-s3-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-s3-sink
    properties:
      accessKey: "The Access Key"
      bucketNameOrArn: "The Bucket Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
```

11.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

11.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-s3-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-s3-sink-binding.yaml
```

11.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-s3-sink -p "sink.accessKey=The Access Key" -p "sink.bucketNameOrArn=The Bucket Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

11.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-s3-sink.kamelet.yaml

CHAPTER 12. AWS S3 SOURCE

Receive data from AWS S3.

12.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-s3-source** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS	string		
bucketNameOrArn *	Bucket Name	The S3 Bucket name or ARN	string		
region *	AWS Region	The AWS region to connect to	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS	string		
autoCreateBucket	Autocreate Bucket	Setting the autocreation of the S3 bucket bucketName.	boolean	**false**	
deleteAfterRead	Auto-delete Objects	Delete objects after consuming them	boolean	**true**	

NOTE

Fields marked with an asterisk (*) are mandatory.

12.2. DEPENDENCIES

At runtime, the **aws-s3-source** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:aws2-s3

12.3. USAGE

This section describes how you can use the **aws-s3-source**.

12.3.1. Knative Source

You can use the **aws-s3-source** Kamelet as a Knative source by binding it to a Knative object.

aws-s3-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-s3-source
    properties:
      accessKey: "The Access Key"
      bucketNameOrArn: "The Bucket Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

12.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

12.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-s3-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f aws-s3-source-binding.yaml
```

12.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-s3-source -p "source.accessKey=The Access Key" -p
"source.bucketNameOrArn=The Bucket Name" -p "source.region=eu-west-1" -p
"source.secretKey=The Secret Key" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

12.3.2. Kafka Source

You can use the **aws-s3-source** Kamelet as a Kafka source by binding it to a Kafka topic.

aws-s3-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-s3-source
    properties:
      accessKey: "The Access Key"
      bucketNameOrArn: "The Bucket Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

12.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

12.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-s3-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f aws-s3-source-binding.yaml
```

12.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind aws-s3-source -p "source.accessKey=The Access Key" -p
"source.bucketNameOrArn=The Bucket Name" -p "source.region=eu-west-1" -p
"source.secretKey=The Secret Key" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

12.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-s3-source.kamelet.yaml

CHAPTER 13. AWS S3 STREAMING UPLOAD SINK

Upload data to AWS S3 in streaming upload mode.

13.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **aws-s3-streaming-upload-sink** Kamelet:

Property	Name	Description	Type	Default	Example
accessKey *	Access Key	The access key obtained from AWS.	string		
bucketNameOrArn *	Bucket Name	The S3 Bucket name or ARN.	string		
keyName *	Key Name	Setting the key name for an element in the bucket through endpoint parameter. In Streaming Upload, with the default configuration, this will be the base for the progressive creation of files.	string		
region *	AWS Region	The AWS region to connect to.	string		**"eu-west-1"**
secretKey *	Secret Key	The secret key obtained from AWS.	string		
autoCreateBucket	Autocreate Bucket	Setting the autocreation of the S3 bucket bucketName.	boolean	**false**	
batchMessageNumber	Batch Message Number	The number of messages composing a batch in streaming upload mode	int	**10**	
batchSize	Batch Size	The batch size (in bytes) in streaming upload mode	int	**1000000**	

Property	Name	Description	Type	Default	Example
namingStrategy	Naming Strategy	The naming strategy to use in streaming upload mode. There are 2 enums and the value can be one of progressive, random	string	**"progressive"**	
restartingPolicy	Restarting Policy	The restarting policy to use in streaming upload mode. There are 2 enums and the value can be one of override, lastPart	string	**"lastPart"**	
streamingUploadMode	Streaming Upload Mode	Setting the Streaming Upload Mode	boolean	**true**	

NOTE

Fields marked with an asterisk (*) are mandatory.

13.2. DEPENDENCIES

At runtime, the **aws-s3-streaming-upload-sink** Kamelet relies upon the presence of the following dependencies:

- camel:aws2-s3
- camel:kamelet

13.3. USAGE

This section describes how you can use the **aws-s3-streaming-upload-sink**.

13.3.1. Knative Sink

You can use the **aws-s3-streaming-upload-sink** Kamelet as a Knative sink by binding it to a Knative object.

aws-s3-streaming-upload-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-streaming-upload-sink-binding
spec:
```

```
source:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: aws-s3-streaming-upload-sink
  properties:
    accessKey: "The Access Key"
    bucketNameOrArn: "The Bucket Name"
    keyName: "The Key Name"
    region: "eu-west-1"
    secretKey: "The Secret Key"
```

13.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

13.3.1.2. Procedure for using the cluster CLI

1. Save the **aws-s3-streaming-upload-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-s3-streaming-upload-sink-binding.yaml
```

13.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel aws-s3-streaming-upload-sink -p "sink.accessKey=The Access Key" -p "sink.bucketNameOrArn=The Bucket Name" -p "sink.keyName=The Key Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

13.3.2. Kafka Sink

You can use the **aws-s3-streaming-upload-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

aws-s3-streaming-upload-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-s3-streaming-upload-sink-binding
spec:
  source:
```

```
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: aws-s3-streaming-upload-sink
    properties:
      accessKey: "The Access Key"
      bucketNameOrArn: "The Bucket Name"
      keyName: "The Key Name"
      region: "eu-west-1"
      secretKey: "The Secret Key"
```

13.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

13.3.2.2. Procedure for using the cluster CLI

1. Save the **aws-s3-streaming-upload-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f aws-s3-streaming-upload-sink-binding.yaml
```

13.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-s3-streaming-upload-sink -p
"sink.accessKey=The Access Key" -p "sink.bucketNameOrArn=The Bucket Name" -p
"sink.keyName=The Key Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

This command creates the KameletBinding in the current namespace on the cluster.

13.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/aws-s3-streaming-upload-sink.kamelet.yaml

CHAPTER 14. CASSANDRA SINK

IMPORTANT

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

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

Send data to a Cassandra Cluster.

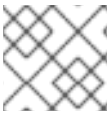
This Kamelet expects the body as JSON Array. The content of the JSON Array will be used as input for the CQL Prepared Statement set in the query parameter.

14.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **cassandra-sink** Kamelet:

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname(s) cassandra server(s). Multiple hosts can be separated by comma.	string		**"localhost"**
connectionPort *	Connection Port	Port number of cassandra server(s)	string		**9042**
keyspace *	Keyspace	Keyspace to use	string		**"customers"**
password *	Password	The password to use for accessing a secured Cassandra Cluster	string		
query *	Query	The query to execute against the Cassandra cluster table	string		
username *	Username	The username to use for accessing a secured Cassandra Cluster	string		

Property	Name	Description	Type	Default	Example
consistency Level	Consistency Level	Consistency level to use. The value can be one of ANY, ONE, TWO, THREE, QUORUM, ALL, LOCAL_QUORUM, EACH_QUORUM, SERIAL, LOCAL_SERIAL, LOCAL_ONE	string	**"ANY"**	



NOTE

Fields marked with an asterisk (*) are mandatory.

14.2. DEPENDENCIES

At runtime, the **cassandra-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:cassandraql

14.3. USAGE

This section describes how you can use the **cassandra-sink**.

14.3.1. Knative Sink

You can use the **cassandra-sink** Kamelet as a Knative sink by binding it to a Knative object.

cassandra-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: cassandra-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
```

```
    name: cassandra-sink
  properties:
    connectionHost: "localhost"
    connectionPort: 9042
    keyspace: "customers"
    password: "The Password"
    query: "The Query"
    username: "The Username"
```

14.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

14.3.1.2. Procedure for using the cluster CLI

1. Save the **cassandra-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f cassandra-sink-binding.yaml
```

14.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel cassandra-sink -p "sink.connectionHost=localhost" -p
sink.connectionPort=9042 -p "sink.keyspace=customers" -p "sink.password=The Password" -p
"sink.query=Query" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

14.3.2. Kafka Sink

You can use the **cassandra-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

cassandra-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: cassandra-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
```

```
  name: cassandra-sink
properties:
  connectionHost: "localhost"
  connectionPort: 9042
  keyspace: "customers"
  password: "The Password"
  query: "The Query"
  username: "The Username"
```

14.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

14.3.2.2. Procedure for using the cluster CLI

1. Save the **cassandra-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f cassandra-sink-binding.yaml
```

14.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic cassandra-sink -p
"sink.connectionHost=localhost" -p sink.connectionPort=9042 -p "sink.keyspace=customers" -p
"sink.password=The Password" -p "sink.query=The Query" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

14.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/cassandra-sink.kamelet.yaml

CHAPTER 15. CASSANDRA SOURCE

IMPORTANT

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

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

Query a Cassandra cluster table.

15.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **cassandra-source** Kamelet:

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname(s) cassandra server(s). Multiple hosts can be separated by comma.	string		**"localhost"**
connectionPort *	Connection Port	Port number of cassandra server(s)	string		**9042**
keyspace *	Keyspace	Keyspace to use	string		**"customers"**
password *	Password	The password to use for accessing a secured Cassandra Cluster	string		
query *	Query	The query to execute against the Cassandra cluster table	string		
username *	Username	The username to use for accessing a secured Cassandra Cluster	string		

Property	Name	Description	Type	Default	Example
consistencyLevel	Consistency Level	Consistency level to use. The value can be one of ANY, ONE, TWO, THREE, QUORUM, ALL, LOCAL_QUORUM, EACH_QUORUM, SERIAL, LOCAL_SERIAL, LOCAL_ONE	string	**"QUORUM"**	
resultStrategy	Result Strategy	The strategy to convert the result set of the query. Possible values are ALL, ONE, LIMIT_10, LIMIT_100...	string	**"ALL"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

15.2. DEPENDENCIES

At runtime, the **cassandra-source** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:cassandraql

15.3. USAGE

This section describes how you can use the **cassandra-source**.

15.3.1. Knative Source

You can use the **cassandra-source** Kamelet as a Knative source by binding it to a Knative object.

cassandra-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: cassandra-source-binding
spec:
  source:
    ref:
      kind: Kamelet
```

```
    apiVersion: camel.apache.org/v1alpha1
    name: cassandra-source
  properties:
    connectionHost: "localhost"
    connectionPort: 9042
    keyspace: "customers"
    password: "The Password"
    query: "The Query"
    username: "The Username"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
```

15.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

15.3.1.2. Procedure for using the cluster CLI

1. Save the **cassandra-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f cassandra-source-binding.yaml
```

15.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind cassandra-source -p "source.connectionHost=localhost" -p source.connectionPort=9042 -p "source.keyspace=customers" -p "source.password=The Password" -p "source.query=The Query" -p "source.username=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

15.3.2. Kafka Source

You can use the **cassandra-source** Kamelet as a Kafka source by binding it to a Kafka topic.

cassandra-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: cassandra-source-binding
spec:
  source:
    ref:
      kind: Kamelet
```

```
    apiVersion: camel.apache.org/v1alpha1
    name: cassandra-source
  properties:
    connectionHost: "localhost"
    connectionPort: 9042
    keyspace: "customers"
    password: "The Password"
    query: "The Query"
    username: "The Username"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

15.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

15.3.2.2. Procedure for using the cluster CLI

1. Save the **cassandra-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f cassandra-source-binding.yaml
```

15.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind cassandra-source -p "source.connectionHost=localhost" -p source.connectionPort=9042 -p "source.keyspace=customers" -p "source.password=The Password" -p "source.query=The Query" -p "source.username=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

15.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/cassandra-source.kamelet.yaml

CHAPTER 16. ELASTICSEARCH INDEX SINK

IMPORTANT

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

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

This sink stores documents into ElasticSearch.

Input data must have JSON format according to the index used.

The Kamelet expects the following headers:

- **indexId** / **ce-indexid**: as the index ID for Elasticsearch

If the header won't be set, the index will be generated by the ES Cluster.

- **indexName** / **ce-indexname**: as the index Name for Elasticsearch

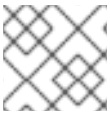
If the header is not set, **camel-k-index-es** will be used as the index name.

16.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **elasticsearch-index-sink** Kamelet:

Property	Name	Description	Type	Default	Example
clusterName *	ElasticSearch Cluster Name	Name of the cluster.	string		**"quickstart"**
hostAddresses *	Host Addresses	Comma separated list with ip:port formatted remote transport addresses to use.	string		**"quickstart-es-http:9200"**
enableSSL	Enable SSL	Do we want to connect using SSL?	boolean	**true**	
indexName	Index in ElasticSearch	The name of the index to act against.	string		**"data"**

Property	Name	Description	Type	Default	Example
password	Password	Password to connect to ElasticSearch.	string		
user	Username	Username to connect to ElasticSearch.	string		



NOTE

Fields marked with an asterisk (*) are mandatory.

16.2. DEPENDENCIES

At runtime, the **elasticsearch-index-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- mvn:org.apache.camel.k:camel-k-kamelet-reify
- camel:elasticsearch-rest
- camel:gson
- camel:bean

16.3. USAGE

This section describes how you can use the **elasticsearch-index-sink**.

16.3.1. Knative Sink

You can use the **elasticsearch-index-sink** Kamelet as a Knative sink by binding it to a Knative object.

elasticsearch-index-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: elasticsearch-index-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

```
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: elasticsearch-index-sink
    properties:
      clusterName: "quickstart"
      hostAddresses: "quickstart-es-http:9200"
```

16.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

16.3.1.2. Procedure for using the cluster CLI

1. Save the **elasticsearch-index-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f elasticsearch-index-sink-binding.yaml
```

16.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel elasticsearch-index-sink -p "sink.clusterName=quickstart" -p
"sink.hostAddresses=quickstart-es-http:9200"
```

This command creates the KameletBinding in the current namespace on the cluster.

16.3.2. Kafka Sink

You can use the **elasticsearch-index-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

elasticsearch-index-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: elasticsearch-index-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: elasticsearch-index-sink
```

```
properties:
  clusterName: "quickstart"
  hostAddresses: "quickstart-es-http:9200"
```

16.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

16.3.2.2. Procedure for using the cluster CLI

1. Save the **elasticsearch-index-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f elasticsearch-index-sink-binding.yaml
```

16.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic elasticsearch-index-sink -p
"sink.clusterName=quickstart" -p "sink.hostAddresses=quickstart-es-http:9200"
```

This command creates the KameletBinding in the current namespace on the cluster.

16.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/elasticsearch-index-sink.kamelet.yaml

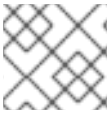
CHAPTER 17. EXTRACT FIELD ACTION

Extract a field from the body

17.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **extract-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
field *	Field	The name of the field to be added	string		



NOTE

Fields marked with an asterisk (*) are mandatory.

17.2. DEPENDENCIES

At runtime, the **extract-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson

17.3. USAGE

This section describes how you can use the **extract-field-action**.

17.3.1. Knative Action

You can use the **extract-field-action** Kamelet as an intermediate step in a Knative binding.

extract-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: extract-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
```

```
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: extract-field-action
    properties:
      field: "The Field"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

17.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

17.3.1.2. Procedure for using the cluster CLI

1. Save the **extract-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f extract-field-action-binding.yaml
```

17.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step extract-field-action -p "step-0.field=The Field"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

17.3.2. Kafka Action

You can use the **extract-field-action** Kamelet as an intermediate step in a Kafka binding.

extract-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: extract-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
```

```
    message: "Hello"
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: extract-field-action
  properties:
    field: "The Field"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

17.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

17.3.2.2. Procedure for using the cluster CLI

1. Save the **extract-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f extract-field-action-binding.yaml
```

17.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step extract-field-action -p "step-0.field=The Field"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

17.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/extract-field-action.kamelet.yaml

CHAPTER 18. FTP SINK

Send data to an FTP Server.

The Kamelet expects the following headers to be set:

- **file** / **ce-file**: as the file name to upload

If the header won't be set the exchange ID will be used as file name.

18.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **ftp-sink** Kamelet:

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname of the FTP server	string		
connectionPort *	Connection Port	Port of the FTP server	string	**21**	
directoryName *	Directory Name	The starting directory	string		
password *	Password	The password to access the FTP server	string		
username *	Username	The username to access the FTP server	string		
fileExist	File Existence	How to behave in case of file already existent. There are 4 enums and the value can be one of Override, Append, Fail or Ignore	string	**"Override"**	
passiveMode	Passive Mode	Sets passive mode connection	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

18.2. DEPENDENCIES

At runtime, the **ftp-sink** Kamelet relies upon the presence of the following dependencies:

- camel:ftp
- camel:core
- camel:kamelet

18.3. USAGE

This section describes how you can use the **ftp-sink**.

18.3.1. Knative Sink

You can use the **ftp-sink** Kamelet as a Knative sink by binding it to a Knative object.

ftp-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ftp-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ftp-sink
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
```

18.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

18.3.1.2. Procedure for using the cluster CLI

1. Save the **ftp-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f ftp-sink-binding.yaml
```

18.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel ftp-sink -p "sink.connectionHost=The Connection Host" -p
"sink.directoryName=The Directory Name" -p "sink.password=The Password" -p "sink.username=The
Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

18.3.2. Kafka Sink

You can use the **ftp-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

ftp-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ftp-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ftp-sink
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
```

18.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

18.3.2.2. Procedure for using the cluster CLI

1. Save the **ftp-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f ftp-sink-binding.yaml
```

18.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic ftp-sink -p "sink.connectionHost=The
Connection Host" -p "sink.directoryName=The Directory Name" -p "sink.password=The Password" -p
"sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

18.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/ftp-sink.kamelet.yaml

CHAPTER 19. FTP SOURCE

Receive data from an FTP Server.

19.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **ftp-source** Kamelet:

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname of the FTP server	string		
connectionPort *	Connection Port	Port of the FTP server	string	**21**	
directoryName *	Directory Name	The starting directory	string		
password *	Password	The password to access the FTP server	string		
username *	Username	The username to access the FTP server	string		
idempotent	Idempotency	Skip already processed files.	boolean	**true**	
passiveMode	Passive Mode	Sets passive mode connection	boolean	**false**	
recursive	Recursive	If a directory, will look for files in all the sub-directories as well.	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

19.2. DEPENDENCIES

At runtime, the **ftp-source** Kamelet relies upon the presence of the following dependencies:

- camel:ftp
- camel:core

- camel:kamelet

19.3. USAGE

This section describes how you can use the **ftp-source**.

19.3.1. Knative Source

You can use the **ftp-source** Kamelet as a Knative source by binding it to a Knative object.

ftp-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ftp-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ftp-source
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

19.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

19.3.1.2. Procedure for using the cluster CLI

1. Save the **ftp-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f ftp-source-binding.yaml
```

19.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind ftp-source -p "source.connectionHost=The Connection Host" -p
"source.directoryName=The Directory Name" -p "source.password=The Password" -p
"source.username=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

19.3.2. Kafka Source

You can use the **ftp-source** Kamelet as a Kafka source by binding it to a Kafka topic.

ftp-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: ftp-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: ftp-source
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

19.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

19.3.2.2. Procedure for using the cluster CLI

1. Save the **ftp-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f ftp-source-binding.yaml
```

19.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind ftp-source -p "source.connectionHost=The Connection Host" -p
"source.directoryName=The Directory Name" -p "source.password=The Password" -p
"source.username=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

19.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/ftp-source.kamelet.yaml

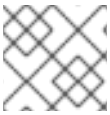
CHAPTER 20. HAS HEADER FILTER ACTION

Filter based on the presence of one header

20.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **has-header-filter-action** Kamelet:

Property	Name	Description	Type	Default	Example
name *	Header Name	The header name to evaluate. The header name must be passed by the source Kamelet. For Knative only, if you are using Cloud Events, you must include the CloudEvent (ce-) prefix in the header name.	string		**"headerName"**



NOTE

Fields marked with an asterisk (*) are mandatory.

20.2. DEPENDENCIES

At runtime, the **has-header-filter-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

20.3. USAGE

This section describes how you can use the **has-header-filter-action**.

20.3.1. Knative Action

You can use the **has-header-filter-action** Kamelet as an intermediate step in a Knative binding.

has-header-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: has-header-filter-action-binding
```

```
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "my-header"
      value: "my-value"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: has-header-filter-action
    properties:
      name: "my-header"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

20.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

20.3.1.2. Procedure for using the cluster CLI

1. Save the **has-header-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f has-header-filter-action-binding.yaml
```

20.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name has-header-filter-action-binding timer-source?message="Hello" --step insert-header-action -p "step-0.name=my-header" -p "step-0.value=my-value" --step has-header-filter-action -p "step-1.name=my-header" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

20.3.2. Kafka Action

You can use the **has-header-filter-action** Kamelet as an intermediate step in a Kafka binding.

has-header-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: has-header-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "my-header"
      value: "my-value"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: has-header-filter-action
    properties:
      name: "my-header"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

20.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

20.3.2.2. Procedure for using the cluster CLI

1. Save the **has-header-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f has-header-filter-action-binding.yaml
```

20.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name has-header-filter-action-binding timer-source?message="Hello" --step insert-header-action -p "step-0.name=my-header" -p "step-0.value=my-value" --step has-header-filter-action -p "step-1.name=my-header" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

20.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/has-header-filter-action.kamelet.yaml

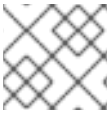
CHAPTER 21. HOIST FIELD ACTION

Wrap data in a single field

21.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **hoist-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
field *	Field	The name of the field that will contain the event	string		



NOTE

Fields marked with an asterisk (*) are mandatory.

21.2. DEPENDENCIES

At runtime, the **hoist-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:core
- camel:jackson
- camel:kamelet

21.3. USAGE

This section describes how you can use the **hoist-field-action**.

21.3.1. Knative Action

You can use the **hoist-field-action** Kamelet as an intermediate step in a Knative binding.

hoist-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: hoist-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
```

```
    message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: hoist-field-action
    properties:
      field: "The Field"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

21.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

21.3.1.2. Procedure for using the cluster CLI

1. Save the **hoist-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f hoist-field-action-binding.yaml
```

21.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step hoist-field-action -p "step-0.field=The Field"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

21.3.2. Kafka Action

You can use the **hoist-field-action** Kamelet as an intermediate step in a Kafka binding.

hoist-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: hoist-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
```

```
  properties:
    message: "Hello"
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: hoist-field-action
  properties:
    field: "The Field"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

21.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

21.3.2.2. Procedure for using the cluster CLI

1. Save the **hoist-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f hoist-field-action-binding.yaml
```

21.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step hoist-field-action -p "step-0.field=The Field"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

21.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/hoist-field-action.kamelet.yaml

CHAPTER 22. HTTP SINK

Forwards an event to a HTTP endpoint

22.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **http-sink** Kamelet:

Property	Name	Description	Type	Default	Example
url *	URL	The URL to send data to	string		**"https://my-service/path"**
method	Method	The HTTP method to use	string	**"POST"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

22.2. DEPENDENCIES

At runtime, the **http-sink** Kamelet relies upon the presence of the following dependencies:

- camel:http
- camel:kamelet
- camel:core

22.3. USAGE

This section describes how you can use the **http-sink**.

22.3.1. Knative Sink

You can use the **http-sink** Kamelet as a Knative sink by binding it to a Knative object.

http-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: http-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
```

```
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: http-sink
    properties:
      url: "https://my-service/path"
```

22.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

22.3.1.2. Procedure for using the cluster CLI

1. Save the **http-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f http-sink-binding.yaml
```

22.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel http-sink -p "sink.url=https://my-service/path"
```

This command creates the KameletBinding in the current namespace on the cluster.

22.3.2. Kafka Sink

You can use the **http-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

http-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: http-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: http-sink
    properties:
      url: "https://my-service/path"
```

22.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

22.3.2.2. Procedure for using the cluster CLI

1. Save the **http-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f http-sink-binding.yaml
```

22.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic http-sink -p "sink.url=https://my-service/path"
```

This command creates the KameletBinding in the current namespace on the cluster.

22.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/http-sink.kamelet.yaml

CHAPTER 23. INSERT FIELD ACTION

Adds a custom field with a constant value to the message in transit

23.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **insert-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
field *	Field	The name of the field to be added	string		
value *	Value	The value of the field	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

23.2. DEPENDENCIES

At runtime, the **insert-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:core
- camel:jackson
- camel:kamelet

23.3. USAGE

This section describes how you can use the **insert-field-action**.

23.3.1. Knative Action

You can use the **insert-field-action** Kamelet as an intermediate step in a Knative binding.

insert-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: insert-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
```

```
    apiVersion: camel.apache.org/v1alpha1
    name: timer-source
  properties:
    message: '{"foo":"John"}'
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: json-deserialize-action
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-field-action
  properties:
    field: "The Field"
    value: "The Value"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
```

23.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

23.3.1.2. Procedure for using the cluster CLI

1. Save the **insert-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f insert-field-action-binding.yaml
```

23.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name insert-field-action-binding timer-source?message='{"foo":"John"}' --step json-deserialize-action --step insert-field-action -p step-1.field='The Field' -p step-1.value='The Value' channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

23.3.2. Kafka Action

You can use the **insert-field-action** Kamelet as an intermediate step in a Kafka binding.

insert-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
```

```
kind: KameletBinding
metadata:
  name: insert-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"foo":"John"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-field-action
    properties:
      field: "The Field"
      value: "The Value"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

23.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

23.3.2.2. Procedure for using the cluster CLI

1. Save the **insert-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f insert-field-action-binding.yaml
```

23.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name insert-field-action-binding timer-source?message='{"foo":"John"}' --step json-deserialize-action --step insert-field-action -p step-1.field='The Field' -p step-1.value='The Value' kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

23.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/insert-field-action.kamelet.yaml

CHAPTER 24. INSERT HEADER ACTION

Adds an header with a constant value to the message in transit

24.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **insert-header-action** Kamelet:

Property	Name	Description	Type	Default	Example
name *	Name	The name of the header to be added. For Knative only, the name of the header requires a CloudEvent (ce-) prefix.	string		
value *	Value	The value of the header	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

24.2. DEPENDENCIES

At runtime, the **insert-header-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

24.3. USAGE

This section describes how you can use the **insert-header-action**.

24.3.1. Knative Action

You can use the **insert-header-action** Kamelet as an intermediate step in a Knative binding.

insert-header-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: insert-header-action-binding
spec:
  source:
    ref:
```

```
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: timer-source
  properties:
    message: "Hello"
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "The Name"
    value: "The Value"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
```

24.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

24.3.1.2. Procedure for using the cluster CLI

1. Save the **insert-header-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f insert-header-action-binding.yaml
```

24.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step insert-header-action -p "step-0.name=The Name" -p
"step-0.value=The Value" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

24.3.2. Kafka Action

You can use the **insert-header-action** Kamelet as an intermediate step in a Kafka binding.

insert-header-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: insert-header-action-binding
spec:
```

```
source:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: timer-source
  properties:
    message: "Hello"
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: insert-header-action
  properties:
    name: "The Name"
    value: "The Value"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

24.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

24.3.2.2. Procedure for using the cluster CLI

1. Save the **insert-header-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f insert-header-action-binding.yaml
```

24.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step insert-header-action -p "step-0.name=The Name" -p
"step-0.value=The Value" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

24.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/insert-header-action.kamelet.yaml

CHAPTER 25. IS TOMBSTONE FILTER ACTION

Filter based on the presence of body or not

25.1. CONFIGURATION OPTIONS

The **is-tombstone-filter-action** Kamelet does not specify any configuration option.

25.2. DEPENDENCIES

At runtime, the **is-tombstone-filter-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

25.3. USAGE

This section describes how you can use the **is-tombstone-filter-action**.

25.3.1. Knative Action

You can use the **is-tombstone-filter-action** Kamelet as an intermediate step in a Knative binding.

is-tombstone-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: is-tombstone-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: is-tombstone-filter-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

25.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

25.3.1.2. Procedure for using the cluster CLI

1. Save the **is-tombstone-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f is-tombstone-filter-action-binding.yaml
```

25.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step is-tombstone-filter-action channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

25.3.2. Kafka Action

You can use the **is-tombstone-filter-action** Kamelet as an intermediate step in a Kafka binding.

is-tombstone-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: is-tombstone-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: is-tombstone-filter-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

25.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

25.3.2.2. Procedure for using the cluster CLI

1. Save the **is-tombstone-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f is-tombstone-filter-action-binding.yaml
```

25.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

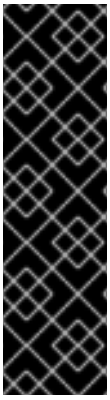
```
kamel bind timer-source?message=Hello --step is-tombstone-filter-action
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

25.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/is-tombstone-filter-action.kamelet.yaml

CHAPTER 26. JIRA SOURCE



IMPORTANT

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

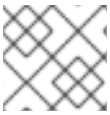
These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process. For more information about the support scope of Red Hat Technology Preview features, see https://access.redhat.com/support/offerings/techpreview.

Receive notifications about new issues from Jira.

26.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jira-source** Kamelet:

Property	Name	Description	Type	Default	Example
jiraUrl *	Jira URL	The URL of your instance of Jira	string		**"http://my_jira.com:8081"**
password *	Password	The password to access Jira	string		
username *	Username	The username to access Jira	string		
jql	JQL	A query to filter issues	string		**"project=MyProject"**



NOTE

Fields marked with an asterisk (*) are mandatory.

26.2. DEPENDENCIES

At runtime, the **jira-source** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:jira

26.3. USAGE

This section describes how you can use the **jira-source**.

26.3.1. Knative Source

You can use the **jira-source** Kamelet as a Knative source by binding it to a Knative object.

jira-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jira-source
    properties:
      jiraUrl: "http://my_jira.com:8081"
      password: "The Password"
      username: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

26.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

26.3.1.2. Procedure for using the cluster CLI

1. Save the **jira-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f jira-source-binding.yaml
```

26.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind jira-source -p "source.jiraUrl=http://my_jira.com:8081" -p "source.password=The Password" -p "source.username=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

26.3.2. Kafka Source

You can use the **jira-source** Kamelet as a Kafka source by binding it to a Kafka topic.

jira-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jira-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jira-source
    properties:
      jiraUrl: "http://my_jira.com:8081"
      password: "The Password"
      username: "The Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

26.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

26.3.2.2. Procedure for using the cluster CLI

1. Save the **jira-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f jira-source-binding.yaml
```

26.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind jira-source -p "source.jiraUrl=http://my_jira.com:8081" -p "source.password=The Password" -p "source.username=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

26.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/jira-source.kamelet.yaml

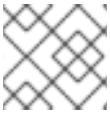
CHAPTER 27. JMS - AMQP 1.0 KAMELET SINK

A Kamelet that can produce events to any AMQP 1.0 compliant message broker using the Apache Qpid JMS client

27.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jms-amqp-10-sink** Kamelet:

Property	Name	Description	Type	Default	Example
destinationName *	Destination Name	The JMS destination name	string		
remoteURI *	Broker URL	The JMS URL	string		**"amqp://my-host:31616"**
destinationType	Destination Type	The JMS destination type (i.e.: queue or topic)	string	**"queue"**	



NOTE

Fields marked with an asterisk (*) are mandatory.

27.2. DEPENDENCIES

At runtime, the **jms-amqp-10-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jms
- camel:kamelet
- mvn:org.apache.qpid:qpid-jms-client:0.55.0

27.3. USAGE

This section describes how you can use the **jms-amqp-10-sink**.

27.3.1. Knative Sink

You can use the **jms-amqp-10-sink** Kamelet as a Knative sink by binding it to a Knative object.

jms-amqp-10-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-amqp-10-sink-binding
spec:
  source:
```

```
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jms-amqp-10-sink
    properties:
      destinationName: "The Destination Name"
      remoteURI: "amqp://my-host:31616"
```

27.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

27.3.1.2. Procedure for using the cluster CLI

1. Save the **jms-amqp-10-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f jms-amqp-10-sink-binding.yaml
```

27.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel jms-amqp-10-sink -p "sink.destinationName=The Destination Name" -p "sink.remoteURI=amqp://my-host:31616"
```

This command creates the KameletBinding in the current namespace on the cluster.

27.3.2. Kafka Sink

You can use the **jms-amqp-10-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

jms-amqp-10-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-amqp-10-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
```

```
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: jms-amqp-10-sink
  properties:
    destinationName: "The Destination Name"
    remoteURI: "amqp://my-host:31616"
```

27.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

27.3.2.2. Procedure for using the cluster CLI

1. Save the **jms-amqp-10-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f jms-amqp-10-sink-binding.yaml
```

27.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic jms-amqp-10-sink -p
"sink.destinationName=The Destination Name" -p "sink.remoteURI=amqp://my-host:31616"
```

This command creates the KameletBinding in the current namespace on the cluster.

27.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/jms-amqp-10-sink.kamelet.yaml

CHAPTER 28. JMS - AMQP 1.0 KAMELET SOURCE

A Kamelet that can consume events from any AMQP 1.0 compliant message broker using the Apache Qpid JMS client

28.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jms-amqp-10-source** Kamelet:

Property	Name	Description	Type	Default	Example
destinationName *	Destination Name	The JMS destination name	string		
remoteURI *	Broker URL	The JMS URL	string		**"amqp://my-host:31616"**
destinationType	Destination Type	The JMS destination type (i.e.: queue or topic)	string	**"queue"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

28.2. DEPENDENCIES

At runtime, the **jms-amqp-10-source** Kamelet relies upon the presence of the following dependencies:

- camel:jms
- camel:kamelet
- mvn:org.apache.qpid:qpid-jms-client:0.55.0

28.3. USAGE

This section describes how you can use the **jms-amqp-10-source**.

28.3.1. Knative Source

You can use the **jms-amqp-10-source** Kamelet as a Knative source by binding it to a Knative object.

jms-amqp-10-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-amqp-10-source-binding
spec:
```

```
source:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: jms-amqp-10-source
  properties:
    destinationName: "The Destination Name"
    remoteURI: "amqp://my-host:31616"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
```

28.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

28.3.1.2. Procedure for using the cluster CLI

1. Save the **jms-amqp-10-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f jms-amqp-10-source-binding.yaml
```

28.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind jms-amqp-10-source -p "source.destinationName=The Destination Name" -p
"source.remoteURI=amqp://my-host:31616" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

28.3.2. Kafka Source

You can use the **jms-amqp-10-source** Kamelet as a Kafka source by binding it to a Kafka topic.

jms-amqp-10-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-amqp-10-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jms-amqp-10-source
```

```
    properties:
      destinationName: "The Destination Name"
      remoteURI: "amqp://my-host:31616"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

28.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

28.3.2.2. Procedure for using the cluster CLI

1. Save the **jms-amqp-10-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f jms-amqp-10-source-binding.yaml
```

28.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind jms-amqp-10-source -p "source.destinationName=The Destination Name" -p
"source.remoteURI=amqp://my-host:31616" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

28.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/jms-amqp-10-source.kamelet.yaml

CHAPTER 29. JMS - IBM MQ KAMELET SINK

A Kamelet that can produce events to an IBM MQ message queue using JMS.

29.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jms-ibm-mq-sink** Kamelet:

Property	Name	Description	Type	Default	Example
channel *	IBM MQ Channel	Name of the IBM MQ Channel	string		
destinationName *	Destination Name	The destination name	string		
password *	Password	Password to authenticate to IBM MQ server	string		
queueManager *	IBM MQ Queue Manager	Name of the IBM MQ Queue Manager	string		
serverName *	IBM MQ Server name	IBM MQ Server name or address	string		
serverPort *	IBM MQ Server Port	IBM MQ Server port	integer	**1414**	
username *	Username	Username to authenticate to IBM MQ server	string		
clientId	IBM MQ Client ID	Name of the IBM MQ Client ID	string		
destinationType	Destination Type	The JMS destination type (queue or topic)	string	**"queue"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

29.2. DEPENDENCIES

At runtime, the **jms-ibm-mq-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jms

- camel:kamelet
- mvn:com.ibm.mq:com.ibm.mq.allclient:9.2.5.0

29.3. USAGE

This section describes how you can use the **jms-ibm-mq-sink**.

29.3.1. Knative Sink

You can use the **jms-ibm-mq-sink** Kamelet as a Knative sink by binding it to a Knative object.

jms-ibm-mq-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-ibm-mq-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
    properties:
      serverName: "10.103.41.245"
      serverPort: "1414"
      destinationType: "queue"
      destinationName: "DEV.QUEUE.1"
      queueManager: QM1
      channel: DEV.APP.SVRCONN
      username: app
      password: passw0rd
```

29.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

29.3.1.2. Procedure for using the cluster CLI

1. Save the **jms-ibm-mq-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f jms-ibm-mq-sink-binding.yaml
```

29.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jms-ibm-mq-sink-binding timer-source?message="Hello IBM MQ!" 'jms-ibm-mq-sink?
serverName=10.103.41.245&serverPort=1414&destinationType=queue&destinationName=DEV.QUEUE.1&queueManager=QM1&channel=DEV.APP.SVRCONN&username=app&password=passw0rd'
```

This command creates the KameletBinding in the current namespace on the cluster.

29.3.2. Kafka Sink

You can use the **jms-ibm-mq-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

jms-ibm-mq-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-ibm-mq-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jms-ibm-mq-sink
    properties:
      serverName: "10.103.41.245"
      serverPort: "1414"
      destinationType: "queue"
      destinationName: "DEV.QUEUE.1"
      queueManager: QM1
      channel: DEV.APP.SVRCONN
      username: app
      password: passw0rd
```

29.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

29.3.2.2. Procedure for using the cluster CLI

1. Save the **jms-ibm-mq-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

   -

```
oc apply -f jms-ibm-mq-sink-binding.yaml
```

29.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind --name jms-ibm-mq-sink-binding timer-source?message="Hello IBM MQ!" 'jms-ibm-mq-sink?
serverName=10.103.41.245&serverPort=1414&destinationType=queue&destinationName=DEV.QUEUE.1&queueManager=QM1&channel=DEV.APP.SVRCONN&username=app&password=passw0rd'
```

This command creates the KameletBinding in the current namespace on the cluster.

29.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/jms-ibm-mq-sink.kamelet.yaml

CHAPTER 30. JMS - IBM MQ KAMELET SOURCE

A Kamelet that can read events from an IBM MQ message queue using JMS.

30.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **jms-ibm-mq-source** Kamelet:

Property	Name	Description	Type	Default	Example
channel *	IBM MQ Channel	Name of the IBM MQ Channel	string		
destinationName *	Destination Name	The destination name	string		
password *	Password	Password to authenticate to IBM MQ server	string		
queueManager *	IBM MQ Queue Manager	Name of the IBM MQ Queue Manager	string		
serverName *	IBM MQ Server name	IBM MQ Server name or address	string		
serverPort *	IBM MQ Server Port	IBM MQ Server port	integer	**1414**	
username *	Username	Username to authenticate to IBM MQ server	string		
clientId	IBM MQ Client ID	Name of the IBM MQ Client ID	string		
destinationType	Destination Type	The JMS destination type (queue or topic)	string	**"queue"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

30.2. DEPENDENCIES

At runtime, the **jms-ibm-mq-source** Kamelet relies upon the presence of the following dependencies:

- camel:jms
- camel:kamelet
- mvn:com.ibm.mq:com.ibm.mq.allclient:9.2.5.0

30.3. USAGE

This section describes how you can use the **jms-ibm-mq-source**.

30.3.1. Knative Source

You can use the **jms-ibm-mq-source** Kamelet as a Knative source by binding it to a Knative object.

jms-ibm-mq-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-ibm-mq-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jms-ibm-mq-source
    properties:
      serverName: "10.103.41.245"
      serverPort: "1414"
      destinationType: "queue"
      destinationName: "DEV.QUEUE.1"
      queueManager: QM1
      channel: DEV.APP.SVRCONN
      username: app
      password: passw0rd
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

30.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

30.3.1.2. Procedure for using the cluster CLI

1. Save the **jms-ibm-mq-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f jms-ibm-mq-source-binding.yaml
```

30.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind --name jms-ibm-mq-source-binding 'jms-ibm-mq-source?
serverName=10.103.41.245&serverPort=1414&destinationType=queue&destinationName=DEV.QUEU
E.1&queueManager=QM1&channel=DEV.APP.SVRCONN&username=app&password=passw0rd'
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

30.3.2. Kafka Source

You can use the **jms-ibm-mq-source** Kamelet as a Kafka source by binding it to a Kafka topic.

jms-ibm-mq-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: jms-ibm-mq-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: jms-ibm-mq-source
    properties:
      serverName: "10.103.41.245"
      serverPort: "1414"
      destinationType: "queue"
      destinationName: "DEV.QUEUE.1"
      queueManager: QM1
      channel: DEV.APP.SVRCONN
      username: app
      password: passw0rd
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

30.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

30.3.2.2. Procedure for using the cluster CLI

1. Save the **jms-ibm-mq-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

   -

```
oc apply -f jms-ibm-mq-source-binding.yaml
```

30.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind --name jms-ibm-mq-source-binding 'jms-ibm-mq-source?
serverName=10.103.41.245&serverPort=1414&destinationType=queue&destinationName=DEV.QUEU
E.1&queueManager=QM1&channel=DEV.APP.SVRCONN&username=app&password=passw0rd'
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

30.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/jms-ibm-mq-source.kamelet.yaml

CHAPTER 31. JSON DESERIALIZE ACTION

Deserialize payload to JSON

31.1. CONFIGURATION OPTIONS

The **json-deserialize-action** Kamelet does not specify any configuration option.

31.2. DEPENDENCIES

At runtime, the **json-deserialize-action** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:core
- camel:jackson

31.3. USAGE

This section describes how you can use the **json-deserialize-action**.

31.3.1. Knative Action

You can use the **json-deserialize-action** Kamelet as an intermediate step in a Knative binding.

json-deserialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: json-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

31.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

31.3.1.2. Procedure for using the cluster CLI

1. Save the **json-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f json-deserialize-action-binding.yaml
```

31.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step json-deserialize-action channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

31.3.2. Kafka Action

You can use the **json-deserialize-action** Kamelet as an intermediate step in a Kafka binding.

json-deserialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: json-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

31.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

31.3.2.2. Procedure for using the cluster CLI

1. Save the **json-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f json-deserialize-action-binding.yaml
```

31.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step json-deserialize-action
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

31.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/json-deserialize-action.kamelet.yaml

CHAPTER 32. JSON SERIALIZE ACTION

Serialize payload to JSON

32.1. CONFIGURATION OPTIONS

The **json-serialize-action** Kamelet does not specify any configuration option.

32.2. DEPENDENCIES

At runtime, the **json-serialize-action** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:core
- camel:jackson

32.3. USAGE

This section describes how you can use the **json-serialize-action**.

32.3.1. Knative Action

You can use the **json-serialize-action** Kamelet as an intermediate step in a Knative binding.

json-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: json-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

32.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

32.3.1.2. Procedure for using the cluster CLI

1. Save the **json-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f json-serialize-action-binding.yaml
```

32.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step json-serialize-action channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

32.3.2. Kafka Action

You can use the **json-serialize-action** Kamelet as an intermediate step in a Kafka binding.

json-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: json-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

32.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

32.3.2.2. Procedure for using the cluster CLI

1. Save the **json-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f json-serialize-action-binding.yaml
```

32.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step json-serialize-action
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

32.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/json-serialize-action.kamelet.yaml

CHAPTER 33. KAFKA SINK

Send data to Kafka topics.

The Kamelet is able to understand the following headers to be set:

- **key** / **ce-key**: as message key
- **partition-key** / **ce-partitionkey**: as message partition key

Both the headers are optional.

33.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **kafka-sink** Kamelet:

Property	Name	Description	Type	Default	Example
bootstrapServers *	Brokers	Comma separated list of Kafka Broker URLs	string		
password *	Password	Password to authenticate to kafka	string		
topic *	Topic Names	Comma separated list of Kafka topic names	string		
user *	Username	Username to authenticate to Kafka	string		
saslMechanism	SASL Mechanism	The Simple Authentication and Security Layer (SASL) Mechanism used.	string	**"PLAIN"**	
securityProtocol	Security Protocol	Protocol used to communicate with brokers. SASL_PLAINTEXT, PLAINTEXT, SASL_SSL and SSL are supported	string	**"SASL_SSL"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

33.2. DEPENDENCIES

At runtime, the `kafka-sink Kamelet relies upon the presence of the following dependencies:

- camel:kafka
- camel:kamelet

33.3. USAGE

This section describes how you can use the **kafka-sink**.

33.3.1. Knative Sink

You can use the **kafka-sink** Kamelet as a Knative sink by binding it to a Knative object.

kafka-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-sink
    properties:
      bootstrapServers: "The Brokers"
      password: "The Password"
      topic: "The Topic Names"
      user: "The Username"
```

33.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

33.3.1.2. Procedure for using the cluster CLI

1. Save the **kafka-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f kafka-sink-binding.yaml
```

33.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel kafka-sink -p "sink.bootstrapServers=The Brokers" -p
"sink.password=The Password" -p "sink.topic=The Topic Names" -p "sink.user=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

33.3.2. Kafka Sink

You can use the **kafka-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

kafka-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-sink
    properties:
      bootstrapServers: "The Brokers"
      password: "The Password"
      topic: "The Topic Names"
      user: "The Username"
```

33.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

33.3.2.2. Procedure for using the cluster CLI

1. Save the **kafka-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f kafka-sink-binding.yaml
```

33.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic kafka-sink -p "sink.bootstrapServers=The Brokers" -p "sink.password=The Password" -p "sink.topic=The Topic Names" -p "sink.user=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

33.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/kafka-sink.kamelet.yaml

CHAPTER 34. KAFKA SOURCE

Receive data from Kafka topics.

34.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **kafka-source** Kamelet:

Property	Name	Description	Type	Default	Example
topic *	Topic Names	Comma separated list of Kafka topic names	string		
bootstrapServers *	Brokers	Comma separated list of Kafka Broker URLs	string		
securityProtocol	Security Protocol	Protocol used to communicate with brokers. SASL_PLAINTEXT, PLAINTEXT, SASL_SSL and SSL are supported	string	**"SASL_SSL"**	
saslMechanism	SASL Mechanism	The Simple Authentication and Security Layer (SASL) Mechanism used.	string	**"PLAIN"**	
user *	Username	Username to authenticate to Kafka	string		
password *	Password	Password to authenticate to kafka	string		
autoCommitEnable	Auto Commit Enable	If true, periodically commit to ZooKeeper the offset of messages already fetched by the consumer.	boolean	**true**	
allowManualCommit	Allow Manual Commit	Whether to allow doing manual commits	boolean	**false**	

Property	Name	Description	Type	Default	Example
autoOffsetReset	Auto Offset Reset	What to do when there is no initial offset. There are 3 enums and the value can be one of latest, earliest, none	string	**"latest"**	
pollOnError	Poll On Error Behavior	What to do if kafka threw an exception while polling for new messages. There are 5 enums and the value can be one of DISCARD, ERROR_HANDLER, RECONNECT, RETRY, STOP	string	**"ERROR_HANDLER"**	
deserializeHeaders	Automatically Deserialize Headers	When enabled the Kamelet source will deserialize all message headers to String representation. The default is **false**.	boolean	**true**	

NOTE

Fields marked with an asterisk (*) are mandatory.

34.2. DEPENDENCIES

At runtime, the `kafka-source Kamelet relies upon the presence of the following dependencies:

- camel:kafka
- camel:kamelet
- camel:core

34.3. USAGE

This section describes how you can use the **kafka-source**.

34.3.1. Knative Source

You can use the **kafka-source** Kamelet as a Knative source by binding it to a Knative object.

kafka-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: kafka-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-source
    properties:
      bootstrapServers: "The Brokers"
      password: "The Password"
      topic: "The Topic Names"
      user: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

34.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

34.3.1.2. Procedure for using the cluster CLI

1. Save the **kafka-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f kafka-source-binding.yaml
```

34.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind kafka-source -p "source.bootstrapServers=The Brokers" -p "source.password=The
Password" -p "source.topic=The Topic Names" -p "source.user=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

34.3.2. Kafka Source

You can use the **kafka-source** Kamelet as a Kafka source by binding it to a Kafka topic.

kafka-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
```

```
  name: kafka-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-source
    properties:
      bootstrapServers: "The Brokers"
      password: "The Password"
      topic: "The Topic Names"
      user: "The Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

34.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

34.3.2.2. Procedure for using the cluster CLI

1. Save the **kafka-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f kafka-source-binding.yaml
```

34.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind kafka-source -p "source.bootstrapServers=The Brokers" -p "source.password=The
Password" -p "source.topic=The Topic Names" -p "source.user=The Username"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

34.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/kafka-source.kamelet.yaml

CHAPTER 35. KAFKA TOPIC NAME MATCHES FILTER ACTION

Filter based on kafka topic value compared to regex

35.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **topic-name-matches-filter-action** Kamelet:

Property	Name	Description	Type	Default	Example
regex *	Regex	The Regex to Evaluate against the Kafka topic name	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

35.2. DEPENDENCIES

At runtime, the **topic-name-matches-filter-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet

35.3. USAGE

This section describes how you can use the **topic-name-matches-filter-action**.

35.3.1. Kafka Action

You can use the **topic-name-matches-filter-action** Kamelet as an intermediate step in a Kafka binding.

topic-name-matches-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: topic-name-matches-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
```

```
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: topic-name-matches-filter-action
  properties:
    regex: "The Regex"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

35.3.1.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

35.3.1.2. Procedure for using the cluster CLI

1. Save the **topic-name-matches-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f topic-name-matches-filter-action-binding.yaml
```

35.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step topic-name-matches-filter-action -p "step-
0.regex=The Regex" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

35.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/topic-name-matches-filter-action.kamelet.yaml

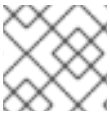
CHAPTER 36. LOG SINK

A sink that logs all data that it receives, useful for debugging purposes.

36.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **log-sink** Kamelet:

Property	Name	Description	Type	Default	Example
showHeaders	Show Headers	Show the headers received	boolean	**false**	
showStreams	Show Streams	Show the stream bodies (they may not be available in following steps)	boolean	**false**	



NOTE

Fields marked with an asterisk (*) are mandatory.

36.2. DEPENDENCIES

At runtime, the **log-sink** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:log

36.3. USAGE

This section describes how you can use the **log-sink**.

36.3.1. Knative Sink

You can use the **log-sink** Kamelet as a Knative sink by binding it to a Knative object.

log-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: log-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
```

```
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
```

36.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

36.3.1.2. Procedure for using the cluster CLI

1. Save the **log-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f log-sink-binding.yaml
```

36.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel log-sink
```

This command creates the KameletBinding in the current namespace on the cluster.

36.3.2. Kafka Sink

You can use the **log-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

log-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: log-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
```

36.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

36.3.2.2. Procedure for using the cluster CLI

1. Save the **log-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f log-sink-binding.yaml
```

36.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic log-sink
```

This command creates the KameletBinding in the current namespace on the cluster.

36.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/log-sink.kamelet.yaml

CHAPTER 37. MARIADB SINK

Send data to a MariaDB Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'

The Kamelet needs to receive as input something like:

'{ "username":"oscerd", "city":"Rome"}'

37.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mariadb-sink** Kamelet:

Property	Name	Description	Type	Default	Example
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured MariaDB Database	string		
query *	Query	The Query to execute against the MariaDB Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured MariaDB Database	string		
serverPort	Server Port	Server Port for the data source	string	**3306**	

NOTE

Fields marked with an asterisk (*) are mandatory.

37.2. DEPENDENCIES

At runtime, the **mariadb-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:org.apache.commons:commons-dbcp2:2.7.0.redhat-00001
- mvn:org.mariadb.jdbc:mariadb-java-client

37.3. USAGE

This section describes how you can use the **mariadb-sink**.

37.3.1. Knative Sink

You can use the **mariadb-sink** Kamelet as a Knative sink by binding it to a Knative object.

mariadb-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mariadb-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mariadb-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

37.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

37.3.1.2. Procedure for using the cluster CLI

1. Save the **mariadb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f mariadb-sink-binding.yaml
```

37.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel mariadb-sink -p "sink.databaseName=The Database Name" -p
"sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES
(:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

37.3.2. Kafka Sink

You can use the **mariadb-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

mariadb-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mariadb-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mariadb-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

37.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

37.3.2.2. Procedure for using the cluster CLI

1. Save the **mariadb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f mariadb-sink-binding.yaml
```

37.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic mariadb-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

37.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/mariadb-sink.kamelet.yaml

CHAPTER 38. MASK FIELDS ACTION

Mask fields with a constant value in the message in transit

38.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mask-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
fields *	Fields	Comma separated list of fields to mask	string		
replacement *	Replacement	Replacement for the fields to be masked	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

38.2. DEPENDENCIES

At runtime, the **mask-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:jackson
- camel:kamelet
- camel:core

38.3. USAGE

This section describes how you can use the **mask-field-action**.

38.3.1. Knative Action

You can use the **mask-field-action** Kamelet as an intermediate step in a Knative binding.

mask-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mask-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
```

```
    apiVersion: camel.apache.org/v1alpha1
    name: timer-source
  properties:
    message: "Hello"
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: mask-field-action
  properties:
    fields: "The Fields"
    replacement: "The Replacement"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
```

38.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

38.3.1.2. Procedure for using the cluster CLI

1. Save the **mask-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f mask-field-action-binding.yaml
```

38.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step mask-field-action -p "step-0.fields=The Fields" -p
"step-0.replacement=The Replacement" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

38.3.2. Kafka Action

You can use the **mask-field-action** Kamelet as an intermediate step in a Kafka binding.

mask-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mask-field-action-binding
spec:
  source:
```

```
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mask-field-action
    properties:
      fields: "The Fields"
      replacement: "The Replacement"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

38.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

38.3.2.2. Procedure for using the cluster CLI

1. Save the **mask-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f mask-field-action-binding.yaml
```

38.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step mask-field-action -p "step-0.fields=The Fields" -p
"step-0.replacement=The Replacement" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

38.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/mask-field-action.kamelet.yaml

CHAPTER 39. MESSAGE TIMESTAMP ROUTER ACTION

Update the topic field as a function of the original topic name and the record's timestamp field.

39.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **message-timestamp-router-action** Kamelet:

Property	Name	Description	Type	Default	Example
timestampKeys *	Timestamp Keys	Comma separated list of Timestamp keys. The timestamp is taken from the first found field.	string		
timestampFormat	Timestamp Format	Format string for the timestamp that is compatible with java.text.SimpleDateFormat.	string	**"yyyyMMdd"**	
timestampKeyFormat	Timestamp Keys Format	Format of the timestamp keys. Possible values are 'timestamp' or any format string for the timestamp that is compatible with java.text.SimpleDateFormat. In case of 'timestamp' the field will be evaluated as milliseconds since 1970, so as a UNIX Timestamp.	string	**"timestamp"**	
topicFormat	Topic Format	Format string which can contain '$[topic]' and '$[timestamp]' as placeholders for the topic and timestamp, respectively.	string	**"topic-$[timestamp]"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

39.2. DEPENDENCIES

At runtime, the **message-timestamp-router-action** Kamelet relies upon the presence of the following dependencies:

- mvn:org.apache.camel.kamelets:camel-kamelets-utils:1.0.0.fuse-800048-redhat-00001
- camel:jackson
- camel:kamelet
- camel:core

39.3. USAGE

This section describes how you can use the **message-timestamp-router-action**.

39.3.1. Knative Action

You can use the **message-timestamp-router-action** Kamelet as an intermediate step in a Knative binding.

message-timestamp-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: message-timestamp-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: message-timestamp-router-action
    properties:
      timestampKeys: "The Timestamp Keys"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

39.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

39.3.1.2. Procedure for using the cluster CLI

1. Save the **message-timestamp-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f message-timestamp-router-action-binding.yaml
```

39.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step message-timestamp-router-action -p "step-
0.timestampKeys=The Timestamp Keys" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

39.3.2. Kafka Action

You can use the **message-timestamp-router-action** Kamelet as an intermediate step in a Kafka binding.

message-timestamp-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: message-timestamp-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: message-timestamp-router-action
    properties:
      timestampKeys: "The Timestamp Keys"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

39.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

39.3.2.2. Procedure for using the cluster CLI

1. Save the **message-timestamp-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f message-timestamp-router-action-binding.yaml
```

39.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step message-timestamp-router-action -p "step-0.timestampKeys=The Timestamp Keys" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

39.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/message-timestamp-router-action.kamelet.yaml

CHAPTER 40. MONGODB SINK

Send documents to MongoDB.

This Kamelet expects a JSON as body.

Properties you can set as headers:

- **db-upsert** / **ce-dbupsert**: if the database should create the element if it does not exist. Boolean value.

40.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mongodb-sink** Kamelet:

Property	Name	Description	Type	Default	Example
collection *	MongoDB Collection	Sets the name of the MongoDB collection to bind to this endpoint.	string		
database *	MongoDB Database	Sets the name of the MongoDB database to target.	string		
hosts *	MongoDB Hosts	Comma separated list of MongoDB Host Addresses in host:port format.	string		
createCollection	Collection	Create collection during initialisation if it doesn't exist.	boolean	**false**	
password	MongoDB Password	User password for accessing MongoDB.	string		
username	MongoDB Username	Username for accessing MongoDB.	string		

Property	Name	Description	Type	Default	Example
writeConcern	Write Concern	Configure the level of acknowledgment requested from MongoDB for write operations, possible values are ACKNOWLEDGED, W1, W2, W3, UNACKNOWLEDGED, JOURNALED, MAJORITY.	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

40.2. DEPENDENCIES

At runtime, the **mongodb-sink** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:mongodb
- camel:jackson

40.3. USAGE

This section describes how you can use the **mongodb-sink**.

40.3.1. Knative Sink

You can use the **mongodb-sink** Kamelet as a Knative sink by binding it to a Knative object.

mongodb-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mongodb-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
```

```
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: mongodb-sink
  properties:
    collection: "The MongoDB Collection"
    database: "The MongoDB Database"
    hosts: "The MongoDB Hosts"
```

40.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

40.3.1.2. Procedure for using the cluster CLI

1. Save the **mongodb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f mongodb-sink-binding.yaml
```

40.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel mongodb-sink -p "sink.collection=The MongoDB Collection" -p
"sink.database=The MongoDB Database" -p "sink.hosts=The MongoDB Hosts"
```

This command creates the KameletBinding in the current namespace on the cluster.

40.3.2. Kafka Sink

You can use the **mongodb-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

mongodb-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mongodb-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mongodb-sink
    properties:
```

```
collection: "The MongoDB Collection"
database: "The MongoDB Database"
hosts: "The MongoDB Hosts"
```

40.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

40.3.2.2. Procedure for using the cluster CLI

1. Save the **mongodb-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f mongodb-sink-binding.yaml
```

40.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic mongodb-sink -p "sink.collection=The MongoDB Collection" -p "sink.database=The MongoDB Database" -p "sink.hosts=The MongoDB Hosts"
```

This command creates the KameletBinding in the current namespace on the cluster.

40.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/mongodb-sink.kamelet.yaml

CHAPTER 41. MONGODB SOURCE

Consume documents from MongoDB.

If the persistentTailTracking option will be enabled, the consumer will keep track of the last consumed message and on the next restart, the consumption will restart from that message. In case of persistentTailTracking enabled, the tailTrackIncreasingField must be provided (by default it is optional).

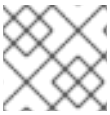
If the persistentTailTracking option won't be enabled, the consumer will consume the whole collection and wait in idle for new documents to consume.

41.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mongodb-source** Kamelet:

Property	Name	Description	Type	Default	Example
collection *	MongoDB Collection	Sets the name of the MongoDB collection to bind to this endpoint.	string		
database *	MongoDB Database	Sets the name of the MongoDB database to target.	string		
hosts *	MongoDB Hosts	Comma separated list of MongoDB Host Addresses in host:port format.	string		
password *	MongoDB Password	User password for accessing MongoDB.	string		
username *	MongoDB Username	Username for accessing MongoDB. The username must be present in the MongoDB's authentication database (authenticationDatabase). By default, the MongoDB authenticationDatabase is 'admin'.	string		

Property	Name	Description	Type	Default	Example
persistentTailTracking	MongoDB Persistent Tail Tracking	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.	boolean	**false**	
tailTrackIncreasingField	MongoDB Tail Track Increasing Field	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.	string		



NOTE

Fields marked with an asterisk (*) are mandatory.

41.2. DEPENDENCIES

At runtime, the **mongodb-source** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:mongodb
- camel:jackson

41.3. USAGE

This section describes how you can use the **mongodb-source**.

41.3.1. Knative Source

You can use the **mongodb-source** Kamelet as a Knative source by binding it to a Knative object.

mongodb-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
```

```
metadata:
  name: mongodb-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mongodb-source
    properties:
      collection: "The MongoDB Collection"
      database: "The MongoDB Database"
      hosts: "The MongoDB Hosts"
      password: "The MongoDB Password"
      username: "The MongoDB Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

41.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

41.3.1.2. Procedure for using the cluster CLI

1. Save the **mongodb-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f mongodb-source-binding.yaml
```

41.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind mongodb-source -p "source.collection=The MongoDB Collection" -p
"source.database=The MongoDB Database" -p "source.hosts=The MongoDB Hosts" -p
"source.password=The MongoDB Password" -p "source.username=The MongoDB Username"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

41.3.2. Kafka Source

You can use the **mongodb-source** Kamelet as a Kafka source by binding it to a Kafka topic.

mongodb-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
```

```
metadata:
  name: mongodb-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mongodb-source
    properties:
      collection: "The MongoDB Collection"
      database: "The MongoDB Database"
      hosts: "The MongoDB Hosts"
      password: "The MongoDB Password"
      username: "The MongoDB Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

41.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

41.3.2.2. Procedure for using the cluster CLI

1. Save the **mongodb-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f mongodb-source-binding.yaml
```

41.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind mongodb-source -p "source.collection=The MongoDB Collection" -p
"source.database=The MongoDB Database" -p "source.hosts=The MongoDB Hosts" -p
"source.password=The MongoDB Password" -p "source.username=The MongoDB Username"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

41.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/mongodb-source.kamelet.yaml

CHAPTER 42. MYSQL SINK

Send data to a MySQL Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'

The Kamelet needs to receive as input something like:

'{ "username":"oscerd", "city":"Rome"}'

42.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **mysql-sink** Kamelet:

Property	Name	Description	Type	Default	Example
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured MySQL Database	string		
query *	Query	The Query to execute against the MySQL Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured MySQL Database	string		
serverPort	Server Port	Server Port for the data source	string	**3306**	

NOTE

Fields marked with an asterisk (*) are mandatory.

42.2. DEPENDENCIES

At runtime, the **mysql-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:org.apache.commons:commons-dbcp2:2.7.0.redhat-00001
- mvn:mysql:mysql-connector-java

42.3. USAGE

This section describes how you can use the **mysql-sink**.

42.3.1. Knative Sink

You can use the **mysql-sink** Kamelet as a Knative sink by binding it to a Knative object.

mysql-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mysql-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mysql-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

42.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

42.3.1.2. Procedure for using the cluster CLI

1. Save the **mysql-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f mysql-sink-binding.yaml
```

42.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel mysql-sink -p "sink.databaseName=The Database Name" -p
"sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES
(:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

42.3.2. Kafka Sink

You can use the **mysql-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

mysql-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: mysql-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: mysql-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

42.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

42.3.2.2. Procedure for using the cluster CLI

1. Save the **mysql-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f mysql-sink-binding.yaml
```

42.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic mysql-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

42.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/mysql-sink.kamelet.yaml

CHAPTER 43. POSTGRESQL SINK

Send data to a PostgreSQL Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'

The Kamelet needs to receive as input something like:

'{ "username":"oscerd", "city":"Rome"}'

43.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **postgresql-sink** Kamelet:

Property	Name	Description	Type	Default	Example
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured PostgreSQL Database	string		
query *	Query	The Query to execute against the PostgreSQL Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured PostgreSQL Database	string		
serverPort	Server Port	Server Port for the data source	string	**5432**	

NOTE

Fields marked with an asterisk (*) are mandatory.

43.2. DEPENDENCIES

At runtime, the **postgresql-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:org.postgresql:postgresql
- mvn:org.apache.commons:commons-dbcp2:2.7.0.redhat-00001

43.3. USAGE

This section describes how you can use the **postgresql-sink**.

43.3.1. Knative Sink

You can use the **postgresql-sink** Kamelet as a Knative sink by binding it to a Knative object.

postgresql-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: postgresql-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: postgresql-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

43.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

43.3.1.2. Procedure for using the cluster CLI

1. Save the **postgresql-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f postgresql-sink-binding.yaml
```

43.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel postgresql-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

43.3.2. Kafka Sink

You can use the **postgresql-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

postgresql-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: postgresql-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: postgresql-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

43.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

43.3.2.2. Procedure for using the cluster CLI

1. Save the **postgresql-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f postgresql-sink-binding.yaml
```

43.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic postgresql-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

43.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/postgresql-sink.kamelet.yaml

CHAPTER 44. PREDICATE FILTER ACTION

Filter based on a JsonPath Expression

44.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **predicate-filter-action** Kamelet:

Property	Name	Description	Type	Default	Example
expression *	Expression	The JsonPath Expression to evaluate, without the external parenthesis. Since this is a filter, the expression will be a negation, this means that if the foo field of the example is equals to John, the message will go ahead, otherwise it will be filtered out.	string		**"@.foo =~ /.*John/"**

NOTE

Fields marked with an asterisk (*) are mandatory.

44.2. DEPENDENCIES

At runtime, the **predicate-filter-action** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:kamelet
- camel:jsonpath

44.3. USAGE

This section describes how you can use the **predicate-filter-action**.

44.3.1. Knative Action

You can use the **predicate-filter-action** Kamelet as an intermediate step in a Knative binding.

predicate-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
```

```
metadata:
  name: predicate-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: predicate-filter-action
    properties:
      expression: "@.foo =~ /.*John/"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

44.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

44.3.1.2. Procedure for using the cluster CLI

1. Save the **predicate-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f predicate-filter-action-binding.yaml
```

44.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step predicate-filter-action -p "step-0.expression=@.foo
=~ /.*John/" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

44.3.2. Kafka Action

You can use the **predicate-filter-action** Kamelet as an intermediate step in a Kafka binding.

predicate-filter-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
```

```
kind: KameletBinding
metadata:
  name: predicate-filter-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: predicate-filter-action
    properties:
      expression: "@.foo =~ /.*John/"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

44.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

44.3.2.2. Procedure for using the cluster CLI

1. Save the **predicate-filter-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f predicate-filter-action-binding.yaml
```

44.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step predicate-filter-action -p "step-0.expression=@.foo
=~ /.*John/" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

44.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/predicate-filter-action.kamelet.yaml

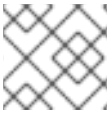
CHAPTER 45. PROTOBUF DESERIALIZE ACTION

Deserialize payload to Protobuf

45.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **protobuf-deserialize-action** Kamelet:

Property	Name	Description	Type	Default	Example
schema *	Schema	The Protobuf schema to use during serialization (as single-line)	string		**"message Person { required string first = 1; required string last = 2; }"**



NOTE

Fields marked with an asterisk (*) are mandatory.

45.2. DEPENDENCIES

At runtime, the **protobuf-deserialize-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson-protobuf

45.3. USAGE

This section describes how you can use the **protobuf-deserialize-action**.

45.3.1. Knative Action

You can use the **protobuf-deserialize-action** Kamelet as an intermediate step in a Knative binding.

protobuf-deserialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: protobuf-deserialize-action-binding
spec:
  source:
    ref:
```

```
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: timer-source
  properties:
    message: '{"first": "John", "last":"Doe"}'
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: json-deserialize-action
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: protobuf-serialize-action
  properties:
    schema: "message Person { required string first = 1; required string last = 2; }"
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: protobuf-deserialize-action
  properties:
    schema: "message Person { required string first = 1; required string last = 2; }"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
```

45.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

45.3.1.2. Procedure for using the cluster CLI

1. Save the **protobuf-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f protobuf-deserialize-action-binding.yaml
```

45.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name protobuf-deserialize-action-binding timer-source?
message='{"first":"John","last":"Doe"}' --step json-deserialize-action --step protobuf-serialize-action -p
step-1.schema='message Person { required string first = 1; required string last = 2; }' --step protobuf-
deserialize-action -p step-2.schema='message Person { required string first = 1; required string last =
2; }' channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

45.3.2. Kafka Action

You can use the **protobuf-deserialize-action** Kamelet as an intermediate step in a Kafka binding.

protobuf-deserialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: protobuf-deserialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first": "John", "last":"Doe"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: protobuf-serialize-action
    properties:
      schema: "message Person { required string first = 1; required string last = 2; }"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: protobuf-deserialize-action
    properties:
      schema: "message Person { required string first = 1; required string last = 2; }"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

45.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

45.3.2.2. Procedure for using the cluster CLI

1. Save the **protobuf-deserialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f protobuf-deserialize-action-binding.yaml
```

45.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name protobuf-deserialize-action-binding timer-source?
message='{"first":"John","last":"Doe"}' --step json-deserialize-action --step protobuf-serialize-action -p
step-1.schema='message Person { required string first = 1; required string last = 2; }' --step protobuf-
deserialize-action -p step-2.schema='message Person { required string first = 1; required string last =
2; }' kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

45.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/protobuf-deserialize-action.kamelet.yaml

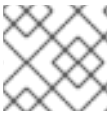
CHAPTER 46. PROTOBUF SERIALIZE ACTION

Serialize payload to Protobuf

46.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **protobuf-serialize-action** Kamelet:

Property	Name	Description	Type	Default	Example
schema *	Schema	The Protobuf schema to use during serialization (as single-line)	string		**"message Person { required string first = 1; required string last = 2; }"**



NOTE

Fields marked with an asterisk (*) are mandatory.

46.2. DEPENDENCIES

At runtime, the **protobuf-serialize-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core
- camel:jackson-protobuf

46.3. USAGE

This section describes how you can use the **protobuf-serialize-action**.

46.3.1. Knative Action

You can use the **protobuf-serialize-action** Kamelet as an intermediate step in a Knative binding.

protobuf-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: protobuf-serialize-action-binding
spec:
  source:
    ref:
```

```
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: timer-source
  properties:
    message: '{"first": "John", "last":"Doe"}'
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: json-deserialize-action
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: protobuf-serialize-action
  properties:
    schema: "message Person { required string first = 1; required string last = 2; }"
sink:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
```

46.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

46.3.1.2. Procedure for using the cluster CLI

1. Save the **protobuf-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f protobuf-serialize-action-binding.yaml
```

46.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name protobuf-serialize-action-binding timer-source?
message='{"first":"John","last":"Doe"}' --step json-deserialize-action --step protobuf-serialize-action -p
step-1.schema='message Person { required string first = 1; required string last = 2; }'
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

46.3.2. Kafka Action

You can use the **protobuf-serialize-action** Kamelet as an intermediate step in a Kafka binding.

protobuf-serialize-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: protobuf-serialize-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: '{"first": "John", "last":"Doe"}'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-deserialize-action
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: protobuf-serialize-action
    properties:
      schema: "message Person { required string first = 1; required string last = 2; }"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

46.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

46.3.2.2. Procedure for using the cluster CLI

1. Save the **protobuf-serialize-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f protobuf-serialize-action-binding.yaml
```

46.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind --name protobuf-serialize-action-binding timer-source?
message='{"first":"John","last":"Doe"}' --step json-deserialize-action --step protobuf-serialize-action -p
step-1.schema='message Person { required string first = 1; required string last = 2; }'
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

46.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/protobuf-serialize-action.kamelet.yaml

CHAPTER 47. REGEX ROUTER ACTION

Update the destination using the configured regular expression and replacement string

47.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **regex-router-action** Kamelet:

Property	Name	Description	Type	Default	Example
regex *	Regex	Regular Expression for destination	string		
replacement *	Replacement	Replacement when matching	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

47.2. DEPENDENCIES

At runtime, the **regex-router-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core

47.3. USAGE

This section describes how you can use the **regex-router-action**.

47.3.1. Knative Action

You can use the **regex-router-action** Kamelet as an intermediate step in a Knative binding.

regex-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: regex-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
```

```
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: regex-router-action
    properties:
      regex: "The Regex"
      replacement: "The Replacement"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

47.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

47.3.1.2. Procedure for using the cluster CLI

1. Save the **regex-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f regex-router-action-binding.yaml
```

47.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step regex-router-action -p "step-0.regex=The Regex" -p
"step-0.replacement=The Replacement" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

47.3.2. Kafka Action

You can use the **regex-router-action** Kamelet as an intermediate step in a Kafka binding.

regex-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: regex-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
```

```
    apiVersion: camel.apache.org/v1alpha1
    name: timer-source
  properties:
    message: "Hello"
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: regex-router-action
  properties:
    regex: "The Regex"
    replacement: "The Replacement"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

47.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

47.3.2.2. Procedure for using the cluster CLI

1. Save the **regex-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f regex-router-action-binding.yaml
```

47.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step regex-router-action -p "step-0.regex=The Regex" -p
"step-0.replacement=The Replacement" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

47.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/regex-router-action.kamelet.yaml

CHAPTER 48. REPLACE FIELD ACTION

Replace field with a different key in the message in transit

48.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **replace-field-action** Kamelet:

Property	Name	Description	Type	Default	Example
disabled *	Disabled	Comma separated list of fields to be disabled	string		
enabled *	Enabled	Comma separated list of fields to be enabled	string		
renames *	Renames	Comma separated list of field with new value to be renamed	string		**"foo:bar,c1:c2"**

NOTE

Fields marked with an asterisk (*) are mandatory.

48.2. DEPENDENCIES

At runtime, the **replace-field-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:core
- camel:jackson
- camel:kamelet

48.3. USAGE

This section describes how you can use the **replace-field-action**.

48.3.1. Knative Action

You can use the **replace-field-action** Kamelet as an intermediate step in a Knative binding.

replace-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
```

```
metadata:
  name: replace-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: replace-field-action
    properties:
      disabled: "The Disabled"
      enabled: "The Enabled"
      renames: "foo:bar,c1:c2"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

48.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

48.3.1.2. Procedure for using the cluster CLI

1. Save the **replace-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f replace-field-action-binding.yaml
```

48.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step replace-field-action -p "step-0.disabled=The
Disabled" -p "step-0.enabled=The Enabled" -p "step-0.renames=foo:bar,c1:c2" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

48.3.2. Kafka Action

You can use the **replace-field-action** Kamelet as an intermediate step in a Kafka binding.

replace-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: replace-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: replace-field-action
    properties:
      disabled: "The Disabled"
      enabled: "The Enabled"
      renames: "foo:bar,c1:c2"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

48.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

48.3.2.2. Procedure for using the cluster CLI

1. Save the **replace-field-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f replace-field-action-binding.yaml
```

48.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step replace-field-action -p "step-0.disabled=The
Disabled" -p "step-0.enabled=The Enabled" -p "step-0.renames=foo:bar,c1:c2"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

48.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/replace-field-action.kamelet.yaml

CHAPTER 49. SALESFORCE SOURCE

Receive updates from Salesforce.

49.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **salesforce-source** Kamelet:

Property	Name	Description	Type	Default	Example
clientId *	Consumer Key	The Salesforce application consumer key	string		
clientSecret *	Consumer Secret	The Salesforce application consumer secret	string		
password *	Password	The Salesforce user password	string		
query *	Query	The query to execute on Salesforce	string		**"SELECT Id, Name, Email, Phone FROM Contact"**
topicName *	Topic Name	The name of the topic/channel to use	string		**"ContactTopic"**
userName *	Username	The Salesforce username	string		
loginUrl	Login URL	The Salesforce instance login URL	string	**"https://login.salesforce.com"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

49.2. DEPENDENCIES

At runtime, the **salesforce-source** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:salesforce
- mvn:org.apache.camel.k:camel-k-kamelet-reify
- camel:kamelet

49.3. USAGE

This section describes how you can use the **salesforce-source**.

49.3.1. Knative Source

You can use the **salesforce-source** Kamelet as a Knative source by binding it to a Knative object.

salesforce-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-source
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      query: "SELECT Id, Name, Email, Phone FROM Contact"
      topicName: "ContactTopic"
      userName: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

49.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

49.3.1.2. Procedure for using the cluster CLI

1. Save the **salesforce-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f salesforce-source-binding.yaml
```

49.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind salesforce-source -p "source.clientId=The Consumer Key" -p "source.clientSecret=The Consumer Secret" -p "source.password=The Password" -p "source.query=SELECT Id, Name, Email,
```

```
Phone FROM Contact" -p "source.topicName=ContactTopic" -p "source.userName=The Username"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

49.3.2. Kafka Source

You can use the **salesforce-source** Kamelet as a Kafka source by binding it to a Kafka topic.

salesforce-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-source
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      query: "SELECT Id, Name, Email, Phone FROM Contact"
      topicName: "ContactTopic"
      userName: "The Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

49.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

49.3.2.2. Procedure for using the cluster CLI

1. Save the **salesforce-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f salesforce-source-binding.yaml
```

49.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind salesforce-source -p "source.clientId=The Consumer Key" -p "source.clientSecret=The Consumer Secret" -p "source.password=The Password" -p "source.query=SELECT Id, Name, Email, Phone FROM Contact" -p "source.topicName=ContactTopic" -p "source.userName=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

49.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/salesforce-source.kamelet.yaml

CHAPTER 50. SFTP SINK

Send data to an SFTP Server.

The Kamelet expects the following headers to be set:

- **file** / **ce-file**: as the file name to upload

If the header won't be set the exchange ID will be used as file name.

50.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **sftp-sink** Kamelet:

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname of the FTP server	string		
connectionPort *	Connection Port	Port of the FTP server	string	**22**	
directoryName *	Directory Name	The starting directory	string		
password *	Password	The password to access the FTP server	string		
username *	Username	The username to access the FTP server	string		
fileExist	File Existence	How to behave in case of file already existent. There are 4 enums and the value can be one of Override, Append, Fail or Ignore	string	**"Override"**	
passiveMode	Passive Mode	Sets passive mode connection	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

50.2. DEPENDENCIES

At runtime, the **sftp-sink** Kamelet relies upon the presence of the following dependencies:

- camel:ftp
- camel:core
- camel:kamelet

50.3. USAGE

This section describes how you can use the **sftp-sink**.

50.3.1. Knative Sink

You can use the **sftp-sink** Kamelet as a Knative sink by binding it to a Knative object.

sftp-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sftp-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sftp-sink
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
```

50.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

50.3.1.2. Procedure for using the cluster CLI

1. Save the **sftp-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the sink by using the following command:

```
oc apply -f sftp-sink-binding.yaml
```

50.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel sftp-sink -p "sink.connectionHost=The Connection Host" -p
"sink.directoryName=The Directory Name" -p "sink.password=The Password" -p "sink.username=The
Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

50.3.2. Kafka Sink

You can use the **sftp-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

sftp-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sftp-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sftp-sink
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
```

50.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

50.3.2.2. Procedure for using the cluster CLI

1. Save the **sftp-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f sftp-sink-binding.yaml
```

50.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic sftp-sink -p "sink.connectionHost=The Connection Host" -p "sink.directoryName=The Directory Name" -p "sink.password=The Password" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

50.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/sftp-sink.kamelet.yaml

CHAPTER 51. SFTP SOURCE

Receive data from an SFTP Server.

51.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **sftp-source** Kamelet:

Property	Name	Description	Type	Default	Example
connectionHost *	Connection Host	Hostname of the SFTP server	string		
connectionPort *	Connection Port	Port of the FTP server	string	**22**	
directoryName *	Directory Name	The starting directory	string		
password *	Password	The password to access the SFTP server	string		
username *	Username	The username to access the SFTP server	string		
idempotent	Idempotency	Skip already processed files.	boolean	**true**	
passiveMode	Passive Mode	Sets passive mode connection	boolean	**false**	
recursive	Recursive	If a directory, will look for files in all the sub-directories as well.	boolean	**false**	

NOTE

Fields marked with an asterisk (*) are mandatory.

51.2. DEPENDENCIES

At runtime, the **sftp-source** Kamelet relies upon the presence of the following dependencies:

- camel:ftp
- camel:core

- camel:kamelet

51.3. USAGE

This section describes how you can use the **sftp-source**.

51.3.1. Knative Source

You can use the **sftp-source** Kamelet as a Knative source by binding it to a Knative object.

sftp-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sftp-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sftp-source
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

51.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

51.3.1.2. Procedure for using the cluster CLI

1. Save the **sftp-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f sftp-source-binding.yaml
```

51.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind sftp-source -p "source.connectionHost=The Connection Host" -p
"source.directoryName=The Directory Name" -p "source.password=The Password" -p
"source.username=The Username" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

51.3.2. Kafka Source

You can use the **sftp-source** Kamelet as a Kafka source by binding it to a Kafka topic.

sftp-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sftp-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sftp-source
    properties:
      connectionHost: "The Connection Host"
      directoryName: "The Directory Name"
      password: "The Password"
      username: "The Username"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

51.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

51.3.2.2. Procedure for using the cluster CLI

1. Save the **sftp-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f sftp-source-binding.yaml
```

51.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind sftp-source -p "source.connectionHost=The Connection Host" -p
"source.directoryName=The Directory Name" -p "source.password=The Password" -p
"source.username=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

51.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/sftp-source.kamelet.yaml

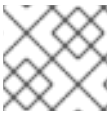
CHAPTER 52. SLACK SOURCE

Receive messages from a Slack channel.

52.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **slack-source** Kamelet:

Property	Name	Description	Type	Default	Example
channel *	Channel	The Slack channel to receive messages from	string		**"#myroom"**
token *	Token	The token to access Slack. A Slack app is needed. This app needs to have channels:history and channels:read permissions. The Bot User OAuth Access Token is the kind of token needed.	string		



NOTE

Fields marked with an asterisk (*) are mandatory.

52.2. DEPENDENCIES

At runtime, the **slack-source** Kamelet relies upon the presence of the following dependencies:

- camel:kamelet
- camel:slack
- camel:jackson

52.3. USAGE

This section describes how you can use the **slack-source**.

52.3.1. Knative Source

You can use the **slack-source** Kamelet as a Knative source by binding it to a Knative object.

slack-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
```

```
kind: KameletBinding
metadata:
  name: slack-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: slack-source
    properties:
      channel: "#myroom"
      token: "The Token"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

52.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

52.3.1.2. Procedure for using the cluster CLI

1. Save the **slack-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the source by using the following command:

```
oc apply -f slack-source-binding.yaml
```

52.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind slack-source -p "source.channel=#myroom" -p "source.token=The Token"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

52.3.2. Kafka Source

You can use the **slack-source** Kamelet as a Kafka source by binding it to a Kafka topic.

slack-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: slack-source-binding
spec:
  source:
```

```
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: slack-source
    properties:
      channel: "#myroom"
      token: "The Token"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

52.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

52.3.2.2. Procedure for using the cluster CLI

1. Save the **slack-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f slack-source-binding.yaml
```

52.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind slack-source -p "source.channel=#myroom" -p "source.token=The Token"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

52.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/slack-source.kamelet.yaml

CHAPTER 53. MICROSOFT SQL SERVER SINK

Send data to a Microsoft SQL Server Database.

This Kamelet expects a JSON as body. The mapping between the JSON fields and parameters is done by key, so if you have the following query:

'INSERT INTO accounts (username,city) VALUES (:#username,:#city)'

The Kamelet needs to receive as input something like:

'{ "username":"oscerd", "city":"Rome"}'

53.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **sqlserver-sink** Kamelet:

Property	Name	Description	Type	Default	Example
databaseName *	Database Name	The Database Name we are pointing	string		
password *	Password	The password to use for accessing a secured SQL Server Database	string		
query *	Query	The Query to execute against the SQL Server Database	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	Server Name	Server Name for the data source	string		**"localhost"**
username *	Username	The username to use for accessing a secured SQL Server Database	string		
serverPort	Server Port	Server Port for the data source	string	**1433**	

NOTE

Fields marked with an asterisk (*) are mandatory.

53.2. DEPENDENCIES

At runtime, the **sqlserver-sink** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:org.apache.commons:commons-dbcp2:2.7.0.redhat-00001
- mvn:com.microsoft.sqlserver:mssql-jdbc:9.2.1.jre11

53.3. USAGE

This section describes how you can use the **sqlserver-sink**.

53.3.1. Knative Sink

You can use the **sqlserver-sink** Kamelet as a Knative sink by binding it to a Knative object.

sqlserver-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sqlserver-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sqlserver-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

53.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

53.3.1.2. Procedure for using the cluster CLI

1. Save the **sqlserver-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f sqlserver-sink-binding.yaml
```

53.3.1.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

```
kamel bind channel:mychannel sqlserver-sink -p "sink.databaseName=The Database Name" -p
"sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES
(:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

53.3.2. Kafka Sink

You can use the **sqlserver-sink** Kamelet as a Kafka sink by binding it to a Kafka topic.

sqlserver-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: sqlserver-sink-binding
spec:
  source:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: sqlserver-sink
    properties:
      databaseName: "The Database Name"
      password: "The Password"
      query: "INSERT INTO accounts (username,city) VALUES (:#username,:#city)"
      serverName: "localhost"
      username: "The Username"
```

53.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

53.3.2.2. Procedure for using the cluster CLI

1. Save the **sqlserver-sink-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the sink by using the following command:

```
oc apply -f sqlserver-sink-binding.yaml
```

53.3.2.3. Procedure for using the Kamel CLI

Configure and run the sink by using the following command:

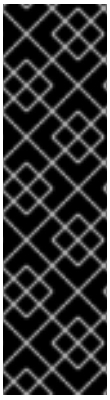
```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic sqlserver-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

This command creates the KameletBinding in the current namespace on the cluster.

53.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/sqlserver-sink.kamelet.yaml

CHAPTER 54. TELEGRAM SOURCE



IMPORTANT

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

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

Receive all messages that people send to your Telegram bot.

To create a bot, contact the @botfather account using the Telegram app.

The source attaches the following headers to the messages:

- **chat-id** / **ce-chatid**: the ID of the chat where the message comes from

54.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **telegram-source** Kamelet:

Property	Name	Description	Type	Default	Example
authorizationToken *	Token	The token to access your bot on Telegram. You you can obtain it from the Telegram @botfather.	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

54.2. DEPENDENCIES

At runtime, the **telegram-source** Kamelet relies upon the presence of the following dependencies:

- camel:jackson
- camel:kamelet
- camel:telegram
- camel:core

54.3. USAGE

This section describes how you can use the **telegram-source**.

54.3.1. Knative Source

You can use the **telegram-source** Kamelet as a Knative source by binding it to a Knative object.

telegram-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: telegram-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: telegram-source
    properties:
      authorizationToken: "The Token"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

54.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

54.3.1.2. Procedure for using the cluster CLI

1. Save the **telegram-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f telegram-source-binding.yaml
```

54.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind telegram-source -p "source.authorizationToken=The Token" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

54.3.2. Kafka Source

You can use the **telegram-source** Kamelet as a Kafka source by binding it to a Kafka topic.

telegram-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: telegram-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: telegram-source
    properties:
      authorizationToken: "The Token"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

54.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

54.3.2.2. Procedure for using the cluster CLI

1. Save the **telegram-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f telegram-source-binding.yaml
```

54.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind telegram-source -p "source.authorizationToken=The Token"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

54.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/telegram-source.kamelet.yaml

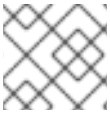
CHAPTER 55. TIMER SOURCE

Produces periodic events with a custom payload.

55.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **timer-source** Kamelet:

Property	Name	Description	Type	Default	Example
message *	Message	The message to generate	string		**"hello world"**
contentType	Content Type	The content type of the message being generated	string	**"text/plain"**	
period	Period	The interval between two events in milliseconds	integer	**1000**	



NOTE

Fields marked with an asterisk (*) are mandatory.

55.2. DEPENDENCIES

At runtime, the **timer-source** Kamelet relies upon the presence of the following dependencies:

- camel:core
- camel:timer
- camel:kamelet

55.3. USAGE

This section describes how you can use the **timer-source**.

55.3.1. Knative Source

You can use the **timer-source** Kamelet as a Knative source by binding it to a Knative object.

timer-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: timer-source-binding
spec:
  source:
```

```
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "hello world"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

55.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

55.3.1.2. Procedure for using the cluster CLI

1. Save the **timer-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f timer-source-binding.yaml
```

55.3.1.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind timer-source -p "source.message=hello world" channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

55.3.2. Kafka Source

You can use the **timer-source** Kamelet as a Kafka source by binding it to a Kafka topic.

timer-source-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: timer-source-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "hello world"
  sink:
```

```
ref:
  kind: KafkaTopic
  apiVersion: kafka.strimzi.io/v1beta1
  name: my-topic
```

55.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

55.3.2.2. Procedure for using the cluster CLI

1. Save the **timer-source-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the source by using the following command:

```
oc apply -f timer-source-binding.yaml
```

55.3.2.3. Procedure for using the Kamel CLI

Configure and run the source by using the following command:

```
kamel bind timer-source -p "source.message=hello world" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

55.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/timer-source.kamelet.yaml

CHAPTER 56. TIMESTAMP ROUTER ACTION

Update the topic field as a function of the original topic name and the record timestamp.

56.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **timestamp-router-action** Kamelet:

Property	Name	Description	Type	Default	Example
timestampFormat	Timestamp Format	Format string for the timestamp that is compatible with java.text.SimpleDateFormat.	string	**"yyyyMMdd"**	
timestampHeaderName	Timestamp Header Name	The name of the header containing a timestamp	string	**"kafka.TIMESTAMP"**	
topicFormat	Topic Format	Format string which can contain '$[topic]' and '$[timestamp]' as placeholders for the topic and timestamp, respectively.	string	**"topic-$[timestamp]"**	

NOTE

Fields marked with an asterisk (*) are mandatory.

56.2. DEPENDENCIES

At runtime, the **timestamp-router-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:kamelet
- camel:core

56.3. USAGE

This section describes how you can use the **timestamp-router-action**.

56.3.1. Knative Action

You can use the **timestamp-router-action** Kamelet as an intermediate step in a Knative binding.

timestamp-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: timestamp-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timestamp-router-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

56.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

56.3.1.2. Procedure for using the cluster CLI

1. Save the **timestamp-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f timestamp-router-action-binding.yaml
```

56.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step timestamp-router-action channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

56.3.2. Kafka Action

You can use the **timestamp-router-action** Kamelet as an intermediate step in a Kafka binding.

timestamp-router-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: timestamp-router-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timestamp-router-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

56.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

56.3.2.2. Procedure for using the cluster CLI

1. Save the **timestamp-router-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f timestamp-router-action-binding.yaml
```

56.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step timestamp-router-action
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

56.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/timestamp-router-action.kamelet.yaml

CHAPTER 57. VALUE TO KEY ACTION

Replace the Kafka record key with a new key formed from a subset of fields in the body

57.1. CONFIGURATION OPTIONS

The following table summarizes the configuration options available for the **value-to-key-action** Kamelet:

Property	Name	Description	Type	Default	Example
fields *	Fields	Comma separated list of fields to be used to form the new key	string		

NOTE

Fields marked with an asterisk (*) are mandatory.

57.2. DEPENDENCIES

At runtime, the **value-to-key-action** Kamelet relies upon the presence of the following dependencies:

- github:openshift-integration.kamelet-catalog:camel-kamelets-utils:kamelet-catalog-1.6-SNAPSHOT
- camel:core
- camel:jackson
- camel:kamelet

57.3. USAGE

This section describes how you can use the **value-to-key-action**.

57.3.1. Knative Action

You can use the **value-to-key-action** Kamelet as an intermediate step in a Knative binding.

value-to-key-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: value-to-key-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
```

```
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: value-to-key-action
    properties:
      fields: "The Fields"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

57.3.1.1. Prerequisite

Make sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

57.3.1.2. Procedure for using the cluster CLI

1. Save the **value-to-key-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.
2. Run the action by using the following command:

```
oc apply -f value-to-key-action-binding.yaml
```

57.3.1.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step value-to-key-action -p "step-0.fields=The Fields"
channel:mychannel
```

This command creates the KameletBinding in the current namespace on the cluster.

57.3.2. Kafka Action

You can use the **value-to-key-action** Kamelet as an intermediate step in a Kafka binding.

value-to-key-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: value-to-key-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
```

```
    name: timer-source
  properties:
    message: "Hello"
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: value-to-key-action
  properties:
    fields: "The Fields"
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

57.3.2.1. Prerequisites

Ensure that you've installed the **AMQ Streams** operator in your OpenShift cluster and created a topic named **my-topic** in the current namespace. Make also sure you have **"Red Hat Integration - Camel K"** installed into the OpenShift cluster you're connected to.

57.3.2.2. Procedure for using the cluster CLI

1. Save the **value-to-key-action-binding.yaml** file to your local drive, and then edit it as needed for your configuration.

2. Run the action by using the following command:

```
oc apply -f value-to-key-action-binding.yaml
```

57.3.2.3. Procedure for using the Kamel CLI

Configure and run the action by using the following command:

```
kamel bind timer-source?message=Hello --step value-to-key-action -p "step-0.fields=The Fields"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

This command creates the KameletBinding in the current namespace on the cluster.

57.4. KAMELET SOURCE FILE

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/value-to-key-action.kamelet.yaml