Red Hat Integration 2022.Q3

Kamelets Reference

Kamelets Reference

Last Updated: 2023-03-07

Red Hat Integration 2022.Q3 Kamelets Reference

Kamelets Reference

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

Kamelets 是可重复使用的路由组件，可隐藏创建连接到外部系统的数据管道的复杂性。

目录

前言

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 AWS DYNAMODB SINK

将数据发送到 AWS DynamoDB 服务。发送的数据将在给定的 AWS DynamoDB 表中插入/更新/删除项。

access Key/Secret Key 是 AWS DynamoDB 服务身份验证的基本方法。这些参数是可选的，因为 Kamelet 还提供以下选项 'useDefaultCredentialsProvider'。

当使用默认凭据提供商时，AWS DynamoDB 客户端将通过此提供程序加载凭据，也不会使用静态凭证。这是无 access key 和 secret key 作为这个 Kamelet 的必要参数的原因。

此 Kamelet 需要一个 JSON 字段作为正文。JSON 字段和表属性值之间的映射由键实现，因此，如果您有输入，如下所示：

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

Kamelet 将在给定的 AWS DynamoDB 表中插入/更新一个项目，并分别设置属性 'username' 和 'city'。请注意，JSON 对象必须包含定义项目的主键值。

1.1. 配置选项

下表总结了 **aws-ddb-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
region *	AWS 区域	要连接的 AWS 区域	字符串		**"eu-west-1"**
表 *	表	要查看的 DynamoDB 表的名称	字符串		
accessKey	访问密钥	从 AWS 获取的访问密钥	字符串		
operation	操作	要执行的操作（一个 PutItem、UpdateItem、DeleteItem）	字符串	**"PutItem"**	**"PutItem"**
overrideEndpoint	endpoint Overwrite	设置端点 URI 的需求。这个选项需要与 uriEndpointOverride 设置结合使用。	布尔值	**false**	
secretKey	机密密钥	从 AWS 获取的 secret 密钥	字符串		
uriEndpointOverride	覆盖端点 URI	设置覆盖端点 URI。这个选项必须与 overrideEndpoint 选项一同使用。	字符串		

属性	名称	描述	类型	默认	示例
useDefaultCredentialsProvider	默认凭证供应商	设置 DynamoDB 客户端是否应该应该通过默认凭据加载凭据，还是期望传递静态凭据。	布尔值	**false**	
writeCapacity	写容量	为将资源写入您的表的置备吞吐量	整数	**1**	

注意

带有星号(*)标记的字段为必填。

1.2. 依赖项

在运行时，**aws-ddb-sink** Kamelet 依赖于以下依赖项：

- mvn:org.apache.camel.kamelets:camel-kamelets-utils:1.8.0
- Camel:core
- Camel:jackson
- camel:aws2-ddb
- camel:kamelet

1.3. 使用方法

这部分论述了如何使用 **aws-ddb-sink**。

1.3.1. Knative Sink

您可以通过将 **aws-ddb-sink** Kamelet 绑定到 Knative 对象来使用 aws-ddb-sink Kamelet 作为 Knative sink。

aws-ddb-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-ddb-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-ddb-sink
  properties:
    region: "eu-west-1"
    table: "The Table"
```

1.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

1.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-ddb-sink-binding.yaml** 文件保存到本地驱动器中，然后根据您的配置需要对其进行编辑。
2. 使用以下命令运行 sink：

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

1.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
kamel bind channel:mychannel aws-ddb-sink -p "sink.region=eu-west-1" -p "sink.table=The Table"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

1.3.2. Kafka Sink

您可以通过将 **aws-ddb-sink** Kamelet 绑定到 Kafka 主题来使用 aws-ddb-sink Kamelet 作为 Kafka sink。

aws-ddb-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: aws-ddb-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-ddb-sink
    properties:
      region: "eu-west-1"
      table: "The Table"
```

1.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

1.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-ddb-sink-binding.yaml** 文件保存到本地驱动器中，然后根据您的配置需要对其进行编辑。
2. 使用以下命令运行 sink：

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

1.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-ddb-sink -p "sink.region=eu-west-1" -p "sink.table=The Table"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

1.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-ddb-sink.kamelet.yaml

第 2 章 AVRO 序列化操作

对 Avro 的序列化有效负载

2.1. 配置选项

下表总结了 **avro-deserialize-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
schema *	模式	在序列化过程中使用 Avro 模式（作为单行，使用 JSON 格式）	字符串		**"{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\": [{\"name\": \"first\", \"type\": \"string\"}, {\"name\": \"name\", \"type\": \"string\"}]}"**
validate	validate	指明是否必须针对 schema 验证内容	布尔值	**true**	

注意

带有星号(*)标记的字段为必填。

2.2. 依赖项

在运行时，av **ro-deserialize-action** Kamelet 依赖于以下依赖项：

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

2.3. 使用方法

本节论述了如何使用 **avro-deserialize-action**。

2.3.1. Knative 操作

您可以使用 **avro-deserialize-action** Kamelet 作为 Knative 绑定中的中间步骤。

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
```

2.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

2.3.1.2. 使用集群 CLI 的步骤

1. 将 **avro-deserialize-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

-

2.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

2.3.2. Kafka 操作

您可以使用 **avro-deserialize-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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
```

2.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

2.3.2.2. 使用集群 CLI 的步骤

1. 将 **avro-deserialize-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

2.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

2.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//avro-deserialize-action.kamelet.yaml

第 3 章 AVRO SERIALIZE ACTION

将有效负载序列化为 Avro

3.1. 配置选项

下表总结了 **avro-serialize-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
schema *	模式	在序列化过程中使用 Avro 模式（作为单行，使用 JSON 格式）	字符串		**"{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\": [{\"name\": \"first\", \"type\": \"string\"}, {\"name\": \"name\", \"type\": \"string\"}]}"**
validate	validate	指明是否必须针对 schema 验证内容	布尔值	**true**	

注意

带有星号(*)标记的字段为必填。

3.2. 依赖项

在运行时，av **ro-serialize-action** Kamelet 依赖于以下依赖项：

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

3.3. 使用方法

本节论述了如何使用 **avro-serialize-action**。

3.3.1. Knative 操作

您可以使用 **avro-serialize-action** Kamelet 作为 Knative 绑定中的中间步骤。

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
```

3.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

3.3.1.2. 使用集群 CLI 的步骤

1. 将 **avro-serialize-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

3.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

3.3.2. Kafka 操作

您可以使用 **avro-serialize-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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
```

3.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

3.3.2.2. 使用集群 CLI 的步骤

1. 将 **avro-serialize-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

3.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

3.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//avro-serialize-action.kamelet.yaml

第 4 章 AWS KINESIS SINK

将数据发送到 AWS Kinesis。

Kamelet 需要以下标头：

- **partition** / **ce-partition**: 设置 Kinesis 分区键

如果没有设置标头，则将使用交换 ID。

Kamelet 也可以识别以下标头：

- **sequence-number** / **ce-sequencenumber**: 以设置序列号

这个标头是可选的。

4.1. 配置选项

下表总结了 **aws-kinesis-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥	字符串		
region *	AWS 区域	要连接的 AWS 区域	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥	字符串		
流 *	流名称	您要访问的 Kinesis 流（需要提前创建）	字符串		

注意

带有星号(*)标记的字段为必填。

4.2. 依赖项

在运行时，**aws-kinesis-sink** Kamelet 依赖于以下依赖项：

- camel:aws2-kinesis
- camel:kamelet

4.3. 使用方法

本节论述了如何使用 **aws-kinesis-sink**。

4.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **aws-kinesis-sink** Kamelet 作为 Knative sink。

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"
```

4.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

4.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-kinesis-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

4.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

4.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，使用 **aws-kinesis-sink** Kamelet 作为 Kafka sink。

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"
```

4.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

4.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-kinesis-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

4.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

4.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-kinesis-sink.kamelet.yaml

第 5 章 AWS KINESIS SOURCE

从 AWS Kinesis 接收数据。

5.1. 配置选项

下表总结了 **aws-kinesis-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥	字符串		
region *	AWS 区域	要连接的 AWS 区域	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥	字符串		
流 *	流名称	您要访问的 Kinesis 流（需要提前创建）	字符串		

注意

带有星号(*)标记的字段为必填。

5.2. 依赖项

在运行时，**aws-kinesis-source** Kamelet 依赖于以下依赖项：

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

5.3. 使用方法

本节论述了如何使用 **aws-kinesis-source**。

5.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，使用 **aws-kinesis-source** Kamelet 作为 Knative 源。

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
```

5.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

5.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-kinesis-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行源：

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

5.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

5.3.2. Kafka Source

您可以通过将其绑定到 Kafka 主题，使用 **aws-kinesis-source** Kamelet 作为 Kafka 源。

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
```

5.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

5.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-kinesis-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行源：

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

5.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

5.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-kinesis-source.kamelet.yaml

第 6 章 AWS LAMBDA SINK

将有效负载发送到 AWS Lambda 功能

6.1. 配置选项

下表总结了 **aws-lambda-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥	字符串		
功能 *	功能名称	Lambda Function name	字符串		
region *	AWS 区域	要连接的 AWS 区域	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥	字符串		

注意

带有星号(*)标记的字段为必填。

6.2. 依赖项

在运行时，**aws-lambda-sink** Kamelet 依赖于以下依赖项：

- camel:kamelet
- camel:aws2-lambda

6.3. 使用方法

本节论述了如何使用 **aws-lambda-sink**。

6.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **aws-lambda-sink** Kamelet 作为 Knative sink。

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"
```

6.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

6.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-lambda-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

6.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

6.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，使用 **aws-lambda-sink** Kamelet 作为 Kafka sink。

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"
```

6.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

6.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-lambda-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

6.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

6.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-lambda-sink.kamelet.yaml

第 7 章 AWS RED HATSHIFT SINK

将数据发送到 AWS Red Hatshift 数据库。

这个 Kamelet 预期 JSON 作为正文。JSON 字段和参数之间的映射由键进行，因此如果您有以下查询：

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

Kamelet 需要作为输入内容接收，如下所示：

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

7.1. 配置选项

下表总结了 **aws-redshift-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
databaseName *	数据库名称	我们指向的数据库名称	字符串		
密码 *	密码	用于访问安全 AWS Redshift 数据库的密码	字符串		
查询 *	查询	要针对 AWS Redshift 数据库执行的查询	字符串		**"INSERT INTO 帐户 (username,city)VALUES(:#username,:#city)"**
serverName *	服务器名称	数据源的服务器名称	字符串		**"localhost"**
用户名 *	用户名	用于访问安全 AWS Redshift 数据库的用户名	字符串		
serverPort	服务器端口	数据源的服务器端口	字符串	**5439**	

注意

带有星号(*)标记的字段为必填。

7.2. 依赖项

在运行时，**aws-redshift-sink** Kamelet 依赖于以下依赖项：

- 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

7.3. 使用方法

本节论述了如何使用 **aws-redshift-sink**。

7.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **aws-redshift-sink** Kamelet 作为 Knative sink。

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"
```

7.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

7.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-redshift-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

7.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

7.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，使用 **aws-redshift-sink** Kamelet 作为 Kafka sink。

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"
```

7.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

7.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-redshift-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

7.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

7.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-redshift-sink.kamelet.yaml

第 8 章 AWS SNS SINK

向 AWS SNS 主题发送消息

8.1. 配置选项

下表总结了 **aws-sns-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥	字符串		
region *	AWS 区域	要连接的 AWS 区域	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥	字符串		
topicNameOrArn *	主题名称	SQS 主题名称或 ARN	字符串		
autoCreateTopic	Autocreate Topic	设置 SNS 主题的自动创建。	布尔值	**false**	

注意

带有星号(*)标记的字段为必填。

8.2. 依赖项

在运行时，**aws-sns-sink** Kamelet 依赖于以下依赖项：

- camel:kamelet
- camel:aws2-sns

8.3. 使用方法

本节论述了如何使用 **aws-sns-sink**。

8.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **aws-sns-sink** Kamelet 作为 Knative sink。

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"
```

8.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

8.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-sns-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

8.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

8.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，使用 **aws-sns-sink** Kamelet 作为 Kafka sink。

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"
```

8.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

8.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-sns-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

8.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

8.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-sns-sink.kamelet.yaml

第 9 章 AWS SQS SINK

向 AWS SQS Queue 发送消息

9.1. 配置选项

下表总结了 **aws-sqs-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥	字符串		
queueNameOrArn *	队列名称	SQS Queue name 或 ARN	字符串		
region *	AWS 区域	要连接的 AWS 区域	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥	字符串		
autoCreateQueue	Autocreate Queue	设置 SQS 队列的自动创建.	布尔值	**false**	

注意

带有星号(*)标记的字段为必填。

9.2. 依赖项

在运行时，**aws-sqs-sink** Kamelet 依赖于以下依赖项：

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

9.3. 使用方法

本节论述了如何使用 **aws-sqs-sink**。

9.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **aws-sqs-sink** Kamelet 作为 Knative sink。

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"
```

9.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

9.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-sqs-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

9.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

9.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，使用 **aws-sqs-sink** Kamelet 作为 Kafka sink。

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"
```

9.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

9.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-sqs-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

9.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

9.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-sqs-sink.kamelet.yaml

第 10 章 AWS SQS 源

从 AWS SQS 接收数据。

10.1. 配置选项

下表总结了 **aws-sqs-source** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥	字符串		
queueNameOrArn *	队列名称	SQS Queue name 或 ARN	字符串		
region *	AWS 区域	要连接的 AWS 区域	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥	字符串		
autoCreateQueue	Autocreate Queue	设置 SQS 队列的自动创建.	布尔值	**false**	
deleteAfterRead	自动删除消息	在消耗信息后删除消息	布尔值	**true**	

注意

带有星号(*)标记的字段为必填。

10.2. 依赖项

在运行时，**aws-sqs-source** Kamelet 依赖于以下依赖项：

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

10.3. 使用方法

本节论述了如何使用 **aws-sqs-source**。

10.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，使用 **aws-sqs-source** Kamelet 作为 Knative 源。

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
```

10.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

10.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-sqs-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行源：

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

10.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

10.3.2. Kafka Source

您可以将 **aws-sqs-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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
```

10.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

10.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-sqs-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行源：

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

10.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

10.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-sqs-source.kamelet.yaml

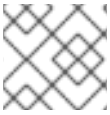
第 11 章 AWS 2 SIMPLE QUEUE SERVICE FIFO SINK

发送消息到 AWS SQS FIFO Queue

11.1. 配置选项

下表总结了 **aws-sqs-fifo-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥	字符串		
queueNameOrArn *	队列名称	SQS Queue name 或 ARN	字符串		
region *	AWS 区域	要连接的 AWS 区域	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥	字符串		
autoCreateQueue	Autocreate Queue	设置 SQS 队列的自动创建.	布尔值	**false**	
contentBasedDeduplication	基于内容的删除重复	使用基于内容的 deduplication（首先在 SQS FIFO FIFO 队列中启用）	布尔值	**false**	



注意

带有星号(*)标记的字段为必填。

11.2. 依赖项

在运行时，**aws-sqs-fifo-sink** Kamelet 依赖于以下依赖项：

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

11.3. 使用方法

本节论述了如何使用 **aws-sqs-fifo-sink**。

11.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **aws-sqs-fifo-sink** Kamelet 作为 Knative 接收器。

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"
```

11.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

11.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-sqs-fifo-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

11.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

11.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，使用 **aws-sqs-fifo-sink** Kamelet 作为 Kafka 接收器。

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"
```

11.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

11.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-sqs-fifo-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

11.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

11.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-sqs-fifo-sink.kamelet.yaml

第 12 章 AWS S3 SINK

将数据上传到 AWS S3。

Kamelet 期望设置以下标头：

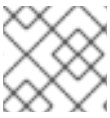
- **文件 / ce-file**: 作为要上传的文件名

如果标头没有设置交换 ID，则将用作文件名。

12.1. 配置选项

下表总结了 **aws-s3-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥。	字符串		
bucketNameOrArn *	存储桶名称	S3 Bucket 名称或 ARN。	字符串		
region *	AWS 区域	要连接的 AWS 区域。	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥。	字符串		
autoCreateBucket	Autocreate Bucket	设置 S3 存储桶 bucketName 的自动记录。	布尔值	**false**	



注意

带有星号(*)标记的字段为必填。

12.2. 依赖项

在运行时，**aws-s3-sink** Kamelet 依赖于以下依赖项：

- camel:aws2-s3
- camel:kamelet

12.3. 使用方法

本节论述了如何使用 **aws-s3-sink**。

12.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **aws-s3-sink** Kamelet 作为 Knative sink。

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"
```

12.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

12.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-s3-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

12.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

12.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，使用 **aws-s3-sink** Kamelet 作为 Kafka sink。

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"
```

12.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

12.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-s3-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

12.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

12.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-s3-sink.kamelet.yaml

第 13 章 AWS S3 源

从 AWS S3 接收数据。

13.1. 配置选项

下表总结了 **aws-s3-source** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥	字符串		
bucketNameOrArn *	存储桶名称	S3 Bucket 名称或 ARN	字符串		
region *	AWS 区域	要连接的 AWS 区域	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥	字符串		
autoCreateBucket	Autocreate Bucket	设置 S3 存储桶 bucketName 的自动记录。	布尔值	**false**	
deleteAfterRead	自动删除对象	在消耗对象后删除对象	布尔值	**true**	

注意

带有星号(*)标记的字段为必填。

13.2. 依赖项

在运行时，**aws-s3-source** Kamelet 依赖于以下依赖项：

- camel:kamelet
- camel:aws2-s3

13.3. 使用方法

本节论述了如何使用 **aws-s3-source**。

13.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，使用 **aws-s3-source** Kamelet 作为 Knative 源。

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
```

13.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

13.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-s3-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

13.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

13.3.2. Kafka Source

您可以将 **aws-s3-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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
```

13.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

13.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-s3-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

13.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

13.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-s3-source.kamelet.yaml

第 14 章 AWS S3 STREAMING UPLOAD SINK

在流上传模式中将数据上传到 AWS S3。

14.1. 配置选项

下表总结了 **aws-s3-streaming-upload-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
accesskey *	访问密钥	从 AWS 获取的访问密钥。	字符串		
bucketNameOrArn *	存储桶名称	S3 Bucket 名称或 ARN。	字符串		
keyName *	键名称	通过 endpoint 参数设置 bucket 中元素的键名称。在流上传中，使用默认配置，这将是最开始创建文件的基础。	字符串		
region *	AWS 区域	要连接的 AWS 区域。	字符串		**"eu-west-1"**
secretKey *	机密密钥	从 AWS 获取的 secret 密钥。	字符串		
autoCreateBucket	Autocreate Bucket	设置 S3 存储桶 bucketName 的自动记录。	布尔值	**false**	
batchMessageNumber	批量消息号	以流上传模式生成批处理的消息数量	int	**10**	
batchSize	批处理大小	批处理大小（以字节为单位）以流传输上传模式	int	**1000000**	
namingStrategy	命名策略	在流传输上传模式中使用的命名策略。有 2 个枚举，值可以是逐渐、随机的	字符串	**"progressive"**	
restartingPolicy	重启策略	在流传输上传模式中使用的重启策略。有 2 个枚举，值可以是 override(lastPart)之一	字符串	**"lastPart"**	

属性	名称	描述	类型	默认	示例
streamingUploadMode	流上传模式	设置流上传模式	布尔值	**true**	

注意

带有星号(*)标记的字段为必填。

14.2. 依赖项

在运行时，**aws-s3-streaming-upload-sink** Kamelet 依赖于以下依赖项：

- camel:aws2-s3
- camel:kamelet

14.3. 使用方法

本节论述了如何使用 **aws-s3-streaming-upload-sink**。

14.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **aws-s3-streaming-upload-sink** Kamelet 作为 Knative 接收器。

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"
```

14.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

14.3.1.2. 使用集群 CLI 的步骤

1. 将 **aws-s3-streaming-upload-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行 sink：

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

14.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

14.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，使用 **aws-s3-streaming-upload-sink** Kamelet 作为 Kafka 接收器。

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"
```

14.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

14.3.2.2. 使用集群 CLI 的步骤

1. 将 **aws-s3-streaming-upload-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。

2. 使用以下命令运行 sink：

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

14.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

14.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//aws-s3-streaming-upload-sink.kamelet.yaml

第 15 章 CASSANDRA SINK

将数据发送到 Cassandra 集群。

这个 Kamelet 预期正文为 JSON Array。JSON Array 的内容将用作查询参数中设置的 CQL 准备语句的输入。

15.1. 配置选项

下表总结了可用于 **cassandra-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
connectionHost *	连接主机	主机名(s)sandssandra server）。可以使用逗号分隔多个主机。	字符串		**"localhost"**
connectionPort *	连接端口	cassandra 服务器的端口号	字符串		**9042**
keyspace *	keyspace	要使用的密钥空间	字符串		**"客户"**
密码 *	密码	用于访问安全 Cassandra 集群的密码	字符串		
查询 *	查询	要对 Cassandra 集群表执行的查询	字符串		
用户名 *	用户名	用于访问安全 Cassandra 集群的用户名	字符串		
consistencyLevel	致性级别	要使用的一致性级别。该值可以是 ANY、ONE、TWO、THREE、QUORUM、ALL、LOCAL_QUORUM、EACH_QUORUM、SERIAL、LOCAL_SERIAL, LOCAL_SERIAL, LOCAL_ONE	字符串	**"任何"**	

注意

带有星号(*)标记的字段为必填。

15.2. 依赖项

在运行时，**cassandra-sink** Kamelet 依赖于以下依赖项：

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

15.3. 使用方法

这部分论述了如何使用 **cassandra-sink**。

15.3.1. Knative Sink

您可以将 **cassandra-sink** Kamelet 用作 Knative sink，方法是将其绑定到 Knative 对象。

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"
```

15.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

15.3.1.2. 使用集群 CLI 的步骤

1. 将 **cassandra-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

15.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

15.3.2. Kafka Sink

您可以将 **cassandra-sink** Kamelet 用作 Kafka sink，方法是将其绑定到 Kafka 主题。

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"
```

15.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

15.3.2.2. 使用集群 CLI 的步骤

1. 将 **cassandra-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

15.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

15.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//cassandra-sink.kamelet.yaml

第 16 章 CASSANDRA 源

查询 Cassandra 集群表。

16.1. 配置选项

下表总结了 **cassandra-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
connectionHost *	连接主机	主机名(s)sandssandra server）。可以使用逗号分隔多个主机。	字符串		**"localhost"**
connectionPort *	连接端口	cassandra 服务器的端口号	字符串		**9042**
keyspace *	keyspace	要使用的密钥空间	字符串		**"客户"**
密码 *	密码	用于访问安全 Cassandra 集群的密码	字符串		
查询 *	查询	要对 Cassandra 集群表执行的查询	字符串		
用户名 *	用户名	用于访问安全 Cassandra 集群的用户名	字符串		
consistencyLevel	致性级别	要使用的一致性级别。该值可以是 ANY、ONE、TWO、THREE、QUORUM、ALL、LOCAL_QUORUM、EACH_QUORUM、SERIAL、LOCAL_SERIAL, LOCAL_SERIAL, LOCAL_ONE	字符串	**"QUORUM"**	
resultStrategy	结果策略	转换查询结果集的策略。可能的值有 ALL、ONE、LIMIT_10、LIMIT_100...	字符串	**"ALL"**	

注意

带有星号(*)标记的字段为必填。

16.2. 依赖项

在运行时，**cassandra-source** Kamelet 依赖于以下依赖项：

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

16.3. 使用方法

这部分论述了如何使用 **cassandra-source**。

16.3.1. Knative Source

您可以将 **cassandra-source** Kamelet 用作 Knative 源，方法是将其绑定到 Knative 对象。

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
```

16.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

16.3.1.2. 使用集群 CLI 的步骤

1. 将 **cassandra-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

16.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

16.3.2. Kafka Source

您可以将 **cassandra-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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
```

16.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

16.3.2.2. 使用集群 CLI 的步骤

1. 将 **cassandra-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

16.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

16.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//cassandra-source.kamelet.yaml

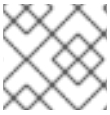
第 17 章 提取字段操作

从正文中提取字段

17.1. 配置选项

下表总结了用于 **extract-field-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
字段 *	字段	要添加的字段名称	字符串		



注意

带有星号(*)标记的字段为必填。

17.2. 依赖项

在运行时，**extract-field-action** Kamelet 依赖于以下依赖项：

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

17.3. 使用方法

本节论述了如何使用 **extract-field-action**。

17.3.1. Knative 操作

您可以使用 **extract-field-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

17.3.1.2. 使用集群 CLI 的步骤

1. 将 **extract-field-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

17.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

17.3.2. Kafka 操作

您可以使用 **extract-field-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

17.3.2.2. 使用集群 CLI 的步骤

1. 将 **extract-field-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

17.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

17.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//extract-field-action.kamelet.yaml

第 18 章 FTP SINK

将数据发送到 FTP 服务器。

Kamelet 期望设置以下标头：

- **文件 / ce-file**: 作为要上传的文件名

如果标头没有设置交换 ID，则将用作文件名。

18.1. 配置选项

下表总结了可用于 **ftp-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
connectionHost *	连接主机	FTP 服务器的主机名	字符串		
connectionPort *	连接端口	FTP 服务器的端口	字符串	**21**	
directoryName *	目录名称	起始目录	字符串		
密码 *	密码	用于访问 FTP 服务器的密码	字符串		
用户名 *	用户名	用于访问 FTP 服务器的用户名	字符串		
fileExist	文件保护	如果文件已存在，如何的行为。有 4 个枚举值，值可以是 Override、Append、Fail 或 Ignore 之一。	字符串	**"Override"**	
passiveMode	被动模式	设置被动模式连接	布尔值	**false**	

注意

带有星号(*)标记的字段为必填。

18.2. 依赖项

在运行时，**ftp-sink** Kamelet 依赖于以下依赖项：

- camel:ftp
- Camel:core

- camel:kamelet

18.3. 使用方法

本节论述了如何使用 **ftp-sink**。

18.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **ftp-sink** Kamelet 作为 Knative sink。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

18.3.1.2. 使用集群 CLI 的步骤

1. 将 **ftp-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

18.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

18.3.2. Kafka Sink

您可以将 **ftp-sink** Kamelet 用作 Kafka 接收器，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

18.3.2.2. 使用集群 CLI 的步骤

1. 将 **ftp-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

18.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

18.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//ftp-sink.kamelet.yaml

第 19 章 FTP 源

从 FTP 服务器接收数据。

19.1. 配置选项

下表总结了可用于 **ftp-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
connectionHost *	连接主机	FTP 服务器的主机名	字符串		
connectionPort *	连接端口	FTP 服务器的端口	字符串	**21**	
directoryName *	目录名称	起始目录	字符串		
密码 *	密码	用于访问 FTP 服务器的密码	字符串		
用户名 *	用户名	用于访问 FTP 服务器的用户名	字符串		
idempotent	幂等性	跳过已处理的文件。	布尔值	**true**	
passiveMode	被动模式	设置被动模式连接	布尔值	**false**	
递归	递归	如果某个目录，也会查找所有子目录中的文件。	布尔值	**false**	

注意

带有星号(*)标记的字段为必填。

19.2. 依赖项

在运行时，**ftp-source** Kamelet 依赖于以下依赖项：

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

19.3. 使用方法

这部分论述了如何使用 **ftp 源**。

19.3.1. Knative Source

您可以将 **ftp-source** Kamelet 用作 Knative 源，方法是将其绑定到 Knative 对象。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

19.3.1.2. 使用集群 CLI 的步骤

1. 将 **ftp-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

19.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

19.3.2. Kafka Source

您可以将 **ftp-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

19.3.2.2. 使用集群 CLI 的步骤

1. 将 **ftp-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

19.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

19.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//ftp-source.kamelet.yaml

第 20 章 带有标题过滤器操作

基于存在一个标头的过滤

20.1. 配置选项

下表总结了可用于 **has-header-filter-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
名称 *	标头名称	要评估的标头名称。源 Kamelet 必须通过标头名称传递。对于 Knative，如果您使用 Cloud Events，您必须在标头名称中包含 CloudEvent(ce-)前缀。	字符串		**"headerName"**

注意

带有星号(*)标记的字段为必填。

20.2. 依赖项

在运行时，with **-header-filter-action** Kamelet 依赖于以下依赖项：

- Camel:core
- camel:kamelet

20.3. 使用方法

本节论述了如何使用 **has-header-filter-action**。

20.3.1. Knative 操作

您可以使用 have **-header-filter-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

20.3.1.2. 使用集群 CLI 的步骤

1. 将 **has-header-filter-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

20.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

20.3.2. Kafka 操作

您可以使用 have **-header-filter-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

20.3.2.2. 使用集群 CLI 的步骤

1. 将 **has-header-filter-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

20.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

20.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//has-header-filter-action.kamelet.yaml

第 21 章 HOIST 字段操作

将数据嵌套在一个字段中

21.1. 配置选项

下表总结了可用于 **hoist-field-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
字段 *	字段	包含事件的字段名称	字符串		

注意

带有星号(*)标记的字段为必填。

21.2. 依赖项

在运行时，**hoist-field-action** Kamelet 依赖于以下依赖项：

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

21.3. 使用方法

这部分论述了如何使用 **hoist-field-action**。

21.3.1. Knative 操作

您可以使用 **hoist-field-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

21.3.1.2. 使用集群 CLI 的步骤

1. 将 **hoist-field-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

21.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

21.3.2. Kafka 操作

您可以使用 **hoist-field-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

21.3.2.2. 使用集群 CLI 的步骤

1. 将 **hoist-field-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

21.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

21.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//hoist-field-action.kamelet.yaml

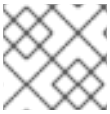
第 22 章 HTTP SINK

将事件转发到 HTTP 端点

22.1. 配置选项

下表总结了 **http-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
url *	URL	将数据发送到的 URL	字符串		**"https://my-service/path"**
method	方法	要使用的 HTTP 方法	字符串	**"POST"**	



注意

带有星号(*)标记的字段为必填。

22.2. 依赖项

在运行时，**http-sink** Kamelet 依赖于以下依赖项：

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

22.3. 使用方法

本节论述了如何使用 **http-sink**。

22.3.1. Knative Sink

您可以将 **http-sink** Kamelet 用作 Knative 接收器，方法是将其绑定到 Knative 对象。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

22.3.1.2. 使用集群 CLI 的步骤

1. 将 **http-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

22.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

22.3.2. Kafka Sink

您可以将 **http-sink** Kamelet 用作 Kafka 接收器，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

22.3.2.2. 使用集群 CLI 的步骤

1. 将 **http-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

22.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

22.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//http-sink.kamelet.yaml

第 23 章 插入字段操作

为传输中的消息添加带有恒定值的自定义字段

23.1. 配置选项

下表总结了可用于 **insert-field-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
字段 *	字段	要添加的字段名称	字符串		
value *	值	字段的值	字符串		

注意

带有星号(*)标记的字段为必填。

23.2. 依赖项

在运行时，**插入-field-action** Kamelet 依赖于以下依赖项：

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

23.3. 使用方法

这部分论述了如何使用 **insert-field-action**。

23.3.1. Knative 操作

您可以使用 **insert-field-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

23.3.1.2. 使用集群 CLI 的步骤

1. 将 **insert-field-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

23.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

23.3.2. Kafka 操作

您可以使用 **insert-field-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

23.3.2.2. 使用集群 CLI 的步骤

1. 将 **insert-field-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

23.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

23.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//insert-field-action.kamelet.yaml

第 24 章 插入标题操作

为传输中的消息添加一个带有恒定值的标头

24.1. 配置选项

下表总结了可用于 **insert-header-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
名称 *	名称	要添加的标头的名称。对于 Knative，标头的名称需要 CloudEvent(ce-)前缀。	字符串		
value *	值	标头的值	字符串		

注意

带有星号(*)标记的字段为必填。

24.2. 依赖项

在运行时，**插入-header-action** Kamelet 依赖于以下依赖项：

- Camel:core
- camel:kamelet

24.3. 使用方法

本节论述了如何使用 **insert-header-action**。

24.3.1. Knative 操作

您可以使用 **insert-header-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

24.3.1.2. 使用集群 CLI 的步骤

1. 将 **insert-header-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

24.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

24.3.2. Kafka 操作

您可以使用 **insert-header-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

24.3.2.2. 使用集群 CLI 的步骤

1. 将 **insert-header-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

24.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

24.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//insert-header-action.kamelet.yaml

第 25 章 是 TOMBSTONE FILTER ACTION

根据正文的存在性或没有过滤

25.1. 配置选项

is-tombstone-filter-action Kamelet 没有指定任何配置选项。

25.2. 依赖项

在运行时，**is-tombstone-filter-action** Kamelet 依赖于以下依赖项：

- Camel:core
- camel:kamelet

25.3. 使用方法

本节论述了如何使用 **is-tombstone-filter-action**。

25.3.1. Knative 操作

您可以使用 **is-tombstone-filter-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

25.3.1.2. 使用集群 CLI 的步骤

1. 将 **is-tombstone-filter-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

25.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

25.3.2. Kafka 操作

您可以使用 **is-tombstone-filter-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

25.3.2.2. 使用集群 CLI 的步骤

1. 将 **is-tombstone-filter-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

25.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

25.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//is-tombstone-filter-action.kamelet.yaml

第 26 章 JIRA SOURCE

从 JIRA 接收有关新问题的通知。

26.1. 配置选项

下表总结了 **jira-source** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
jiraUrl *	Jira URL	JIRAs 实例的 URL	字符串		**"http://my_jira.com:8081"**
密码 *	密码	访问 JIRA 的密码	字符串		
用户名 *	用户名	访问 JIRA 的用户名	字符串		
jql	JQL	用于过滤问题的查询	字符串		**"project=MyProject"**

注意

带有星号(*)标记的字段为必填。

26.2. 依赖项

在运行时，**jira-source** Kamelet 依赖于以下依赖项：

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

26.3. 使用方法

本节论述了如何使用 **jira-source**。

26.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，将 **jira-source** Kamelet 用作 Knative 源。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

26.3.1.2. 使用集群 CLI 的步骤

1. 将 **jira-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

26.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

26.3.2. Kafka Source

您可以将 **jira-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

26.3.2.2. 使用集群 CLI 的步骤

1. 将 **jira-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

26.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

26.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jira-source.kamelet.yaml

第 27 章 JMS - AMQP 1.0 KAMELET SINK

Kamelet，它可使用 Apache Qpid JMS 客户端向任何 AMQP 1.0 兼容消息代理生成事件

27.1. 配置选项

下表总结了可用于 **jms-amqp-10-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
destinationName *	目的地名称	JMS 目的地名称	字符串		
remoteURI *	代理 URL	JMS URL	字符串		**"amqp://my-host:31616"**
destinationType	目的地类型	JMS 目的地类型（例如：队列或主题）	字符串	**"queue"**	

注意

带有星号(*)标记的字段为必填。

27.2. 依赖项

在运行时，**jms-amqp-10-sink** Kamelet 依赖于以下依赖项：

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

27.3. 使用方法

本节介绍如何使用 **jms-amqp-10-sink**。

27.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，将 **jms-amqp-10-sink** Kamelet 用作 Knative 接收器。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

27.3.1.2. 使用集群 CLI 的步骤

1. 将 **jms-amqp-10-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

27.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

27.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，将 **jms-amqp-10-sink** Kamelet 用作 Kafka 接收器。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

27.3.2.2. 使用集群 CLI 的步骤

1. 将 **jms-amqp-10-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

27.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

27.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jms-amqp-10-sink.kamelet.yaml

第 28 章 JMS - AMQP 1.0 KAMELET SOURCE

Kamelet，它可以使用 Apache Qpid JMS 客户端使用任何 AMQP 1.0 兼容消息代理的事件

28.1. 配置选项

下表总结了用于 **jms-amqp-10-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
destinationName *	目的地名称	JMS 目的地名称	字符串		
remoteURI *	代理 URL	JMS URL	字符串		**"amqp://my-host:31616"**
destinationType	目的地类型	JMS 目的地类型（例如：队列或主题）	字符串	**"queue"**	

注意

带有星号(*)标记的字段为必填。

28.2. 依赖项

在运行时，**jms-amqp-10-source** Kamelet 依赖于以下依赖项：

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

28.3. 使用方法

本节介绍如何使用 **jms-amqp-10-source**。

28.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，使用 **jms-amqp-10-source** Kamelet 作为 Knative 源。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

28.3.1.2. 使用集群 CLI 的步骤

1. 将 **jms-amqp-10-source-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

28.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

28.3.2. Kafka Source

您可以通过将其绑定到 Kafka 主题，使用 **jms-amqp-10-source** Kamelet 作为 Kafka 源。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

28.3.2.2. 使用集群 CLI 的步骤

1. 将 **jms-amqp-10-source-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

28.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

28.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jms-amqp-10-source.kamelet.yaml

第 29 章 JMS - IBM MQ KAMELET SINK

一个 Kamelet，可以使用 JMS 向 IBM MQ 消息队列生成事件。

29.1. 配置选项

下表总结了用于 **jms-ibm-mq-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
频道 *	IBM MQ 频道	IBM MQ 频道的名称	字符串		
destinationName *	目的地名称	目的地名称	字符串		
密码 *	密码	用于向 IBM MQ 服务器进行身份验证的密码	字符串		
queueManager *	IBM MQ Queue Manager	IBM MQ Queue Manager 的名称	字符串		
serverName *	IBM MQ 服务器名称	IBM MQ 服务器名称或地址	字符串		
serverPort *	IBM MQ 服务器端口	IBM MQ 服务器端口	整数	**1414**	
用户名 *	用户名	对 IBM MQ 服务器进行身份验证的用户名	字符串		
clientId	IBM MQ 客户端 ID	IBM MQ 客户端 ID 的名称	字符串		
destinationType	目的地类型	JMS 目标类型（队列或主题）	字符串	**"queue"**	

注意

带有星号(*)标记的字段为必填。

29.2. 依赖项

在运行时，**jms-ibm-mq-sink** Kamelet 依赖于以下依赖项：

- Camel:jms
- camel:kamelet

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

29.3. 使用方法

本节介绍如何使用 **jms-ibm-mq-sink**。

29.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **jms-ibm-mq-sink** Kamelet 作为 Knative sink。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

29.3.1.2. 使用集群 CLI 的步骤

1. 将 **jms-ibm-mq-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

29.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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'
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

29.3.2. Kafka Sink

您可以通过绑定到 Kafka 主题，使用 **jms-ibm-mq-sink** Kamelet 作为 Kafka sink。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

29.3.2.2. 使用集群 CLI 的步骤

1. 将 **jms-ibm-mq-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

29.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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'
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

29.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jms-ibm-mq-sink.kamelet.yaml

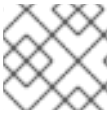
第 30 章 JMS - IBM MQ KAMELET SOURCE

一个 Kamelet，可以使用 JMS 从 IBM MQ 消息队列读取事件。

30.1. 配置选项

下表总结了用于 **jms-ibm-mq-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
频道 *	IBM MQ 频道	IBM MQ 频道的名称	字符串		
destinationName *	目的地名称	目的地名称	字符串		
密码 *	密码	用于向 IBM MQ 服务器进行身份验证的密码	字符串		
queueManager *	IBM MQ Queue Manager	IBM MQ Queue Manager 的名称	字符串		
serverName *	IBM MQ 服务器名称	IBM MQ 服务器名称或地址	字符串		
serverPort *	IBM MQ 服务器端口	IBM MQ 服务器端口	整数	**1414**	
用户名 *	用户名	对 IBM MQ 服务器进行身份验证的用户名	字符串		
clientId	IBM MQ 客户端 ID	IBM MQ 客户端 ID 的名称	字符串		
destinationType	目的地类型	JMS 目标类型（队列或主题）	字符串	**"queue"**	



注意

带有星号(*)标记的字段为必填。

30.2. 依赖项

在运行时，**jms-ibm-mq-source** Kamelet 依赖于以下依赖项：

- Camel:jms
- camel:kamelet

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

30.3. 使用方法

本节介绍如何使用 **jms-ibm-mq-source**。

30.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，使用 **jms-ibm-mq-source** Kamelet 作为 Knative 源。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

30.3.1.2. 使用集群 CLI 的步骤

1. 将 **jms-ibm-mq-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

30.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

30.3.2. Kafka Source

您可以将 **jms-ibm-mq-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

30.3.2.2. 使用集群 CLI 的步骤

1. 将 **jms-ibm-mq-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

30.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

30.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//jms-ibm-mq-source.kamelet.yaml

第 31 章 JSON 序列化操作

对 JSON 进行序列化有效负载

31.1. 配置选项

json-deserialize-action Kamelet 没有指定任何配置选项。

31.2. 依赖项

在运行时，**json-deserialize-action** Kamelet 依赖于以下依赖项：

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

31.3. 使用方法

这部分论述了如何使用 **json 反序列化-action**。

31.3.1. Knative 操作

您可以使用 **json-deserialize-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

31.3.1.2. 使用集群 CLI 的步骤

1. 将 **json-deserialize-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

31.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

31.3.2. Kafka 操作

您可以使用 **json-deserialize-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

31.3.2.2. 使用集群 CLI 的步骤

1. 将 **json-deserialize-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

31.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

31.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//json-deserialize-action.kamelet.yaml

第 32 章 JSON 串行操作

将有效负载化为 JSON

32.1. 配置选项

json-serialize-action Kamelet 没有指定任何配置选项。

32.2. 依赖项

在运行时，**json-serialize-action** Kamelet 依赖于以下依赖项：

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

32.3. 使用方法

这部分论述了如何使用 **json-serialize-action**。

32.3.1. Knative 操作

您可以使用 **json-serialize-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

32.3.1.2. 使用集群 CLI 的步骤

1. 将 **json-serialize-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

32.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

32.3.2. Kafka 操作

您可以使用 **json-serialize-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

32.3.2.2. 使用集群 CLI 的步骤

1. 将 **json-serialize-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

32.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

32.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//json-serialize-action.kamelet.yaml

第 33 章 KAFKA SINK

将数据发送到 Kafka 主题。

Kamelet 能够理解要设置以下标头：

- **密钥 / ce-key**: 作为消息键
- **partition-key / ce-partitionkey** ：作为消息分区密钥

标头都是可选的。

33.1. 配置选项

下表总结了可用于 **kafka-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
bootstrapServers *	代理（Broker）	Kafka Broker URL 的逗号分隔列表	字符串		
密码 *	密码	用于对 kafka 进行身份验证的密码	字符串		
主题 *	主题名称	Kafka 主题名称的逗号分隔列表	字符串		
用户 *	用户名	向 Kafka 进行身份验证的用户名	字符串		
saslMechanism	SASL 机制	使用简单身份验证和安全层(SASL)机制。	字符串	**"说明"**	
securityProtocol	安全协议	用于与代理通信的协议。支持 SASL_PLAINTEXT, PLAINTEXT, SASL_SSL 和 SSL	字符串	**"SASL_SSL"**	

注意

带有星号(*)标记的字段为必填。

33.2. 依赖项

在运行时，"kafka-sink Kamelet 依赖于以下依赖项：

- camel:kafka
- camel:kamelet

33.3. 使用方法

本节论述了如何使用 **kafka-sink**。

33.3.1. Knative Sink

您可以将 **kafka-sink** Kamelet 用作 Knative sink，方法是将其绑定到 Knative 对象。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

33.3.1.2. 使用集群 CLI 的步骤

1. 将 **kafka-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

33.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

33.3.2. Kafka Sink

您可以将 **kafka-sink** Kamelet 用作 Kafka sink，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

33.3.2.2. 使用集群 CLI 的步骤

1. 将 **kafka-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

33.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

33.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//kafka-sink.kamelet.yaml

第 34 章 KAFKA SOURCE

从 Kafka 主题接收数据。

34.1. 配置选项

下表总结了 **kafka-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
主题 *	主题名称	Kafka 主题名称的逗号分隔列表	字符串		
bootstrapServers *	代理（Broker）	Kafka Broker URL 的逗号分隔列表	字符串		
securityProtocol	安全协议	用于与代理通信的协议。支持 SASL_PLAINTEXT, PLAINTEXT, SASL_SSL 和 SSL	字符串	**"SASL_SSL"**	
saslMechanism	SASL 机制	使用简单身份验证和安全层(SASL)机制。	字符串	**"说明"**	
用户 *	用户名	向 Kafka 进行身份验证的用户名	字符串		
密码 *	密码	用于对 kafka 进行身份验证的密码	字符串		
autoCommitEnable	自动提交启用	如果为 true，请定期提交到由消费者获取的消息偏移。	布尔值	**true**	
allowManualCommit	允许手动提交	是否允许手动提交	布尔值	**false**	
autoOffsetReset	自动重置	没有初始偏移时要做什么。有 3 个枚举，值可以是最新的、最早的、无数个	字符串	**"latest"**	
pollOnError	轮询 On Errorhavior	如果 kafka 异常在轮询新消息时要做什么。有 5 个枚举，值可以是 DISCARD、ERROR_HANDLER、RECONNECT、RETRY、STOP	字符串	**"ERROR_HANDLER"**	

属性	名称	描述	类型	默认	示例
deserializeHeaders	自动对标头进行反序列化标头	启用 Kamelet 源后，将把所有消息标头序列化为 String representation。默认值为 **false**。	布尔值	**true**	

注意

带有星号(*)标记的字段为必填。

34.2. 依赖项

在运行时，"kafka-source Kamelet 依赖于以下依赖项：

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

34.3. 使用方法

本节论述了如何使用 **kafka-source**。

34.3.1. Knative Source

您可以将 **kafka-source** Kamelet 用作 Knative 源，方法是将其绑定到 Knative 对象。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

34.3.1.2. 使用集群 CLI 的步骤

1. 将 **kafka-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行源：

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

34.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

34.3.2. Kafka Source

您可以将 **kafka-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

34.3.2.2. 使用集群 CLI 的步骤

1. 将 **kafka-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行源：

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

34.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

34.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//kafka-source.kamelet.yaml

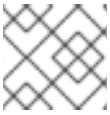
第 35 章 KAFKA TOPIC NAME MATCHES FILTER ACTION

基于 kafka 主题值进行过滤，与 regex 进行比较

35.1. 配置选项

下表总结了可用于 **topic-name-matches-filter-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
regex *	正则表达式	Regex to Evaluate 针对 Kafka 主题名称	字符串		



注意

带有星号(*)标记的字段为必填。

35.2. 依赖项

在运行时，**主题-name-matches-filter-action** Kamelet 依赖于以下依赖项：

- Camel:core
- camel:kamelet

35.3. 使用方法

这部分论述了如何使用 **topic-name-matches-filter-action**。

35.3.1. Kafka 操作

您可以使用 **topic-name-matches-filter-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

35.3.1.2. 使用集群 CLI 的步骤

1. 将 **topic-name-matches-filter-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

35.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

35.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//topic-name-matches-filter-action.kamelet.yaml

第 36 章 LOG SINK

记录接收所有数据的接收器(sink)，可用于调试目的。

36.1. 配置选项

下表总结了 **log-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
showHeaders	显示标头	显示收到的标头	布尔值	**false**	
showStreams	显示流	显示流正文（后续步骤中可能不可用）	布尔值	**false**	

注意

带有星号(*)标记的字段为必填。

36.2. 依赖项

在运行时，**log-sink** Kamelet 依赖于以下依赖项：

- camel:kamelet
- Camel:log

36.3. 使用方法

本节论述了如何使用 **log-sink**。

36.3.1. Knative Sink

您可以将 **log-sink** Kamelet 用作 Knative sink，方法是将其绑定到 Knative 对象。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

36.3.1.2. 使用集群 CLI 的步骤

1. 将 **log-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

36.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

36.3.2. Kafka Sink

您可以将 **log-sink** Kamelet 用作 Kafka 接收器，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

36.3.2.2. 使用集群 CLI 的步骤

1. 将 **log-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

36.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

36.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//log-sink.kamelet.yaml

第 37 章 MARIADB SINK

将数据发送到 MariaDB 数据库。

这个 Kamelet 预期 JSON 作为正文。JSON 字段和参数之间的映射由键进行，因此如果您有以下查询：

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

Kamelet 需要作为输入内容接收，如下所示：

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

37.1. 配置选项

下表总结了 **mariadb-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
databaseName *	数据库名称	我们指向的数据库名称	字符串		
密码 *	密码	用于访问安全 MariaDB 数据库的密码	字符串		
查询 *	查询	对 MariaDB 数据库执行的查询	字符串		**"INSERT INTO 帐户 (username,city)VALUES(:#username,:#city)"**
serverName *	服务器名称	数据源的服务器名称	字符串		**"localhost"**
用户名 *	用户名	用于访问安全 MariaDB 数据库的用户名	字符串		
serverPort	服务器端口	数据源的服务器端口	字符串	**3306**	

注意

带有星号(*)标记的字段为必填。

37.2. 依赖项

在运行时，**mariadb-sink** Kamelet 依赖于以下依赖项：

- 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. 使用方法

本节论述了如何使用 **mariadb-sink**。

37.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **mariadb-sink** Kamelet 作为 Knative sink。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

37.3.1.2. 使用集群 CLI 的步骤

1. 将 **mariadb-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

37.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

37.3.2. Kafka Sink

您可以将 **mariadb-sink** Kamelet 用作 Kafka 接收器，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

37.3.2.2. 使用集群 CLI 的步骤

1. 将 **mariadb-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

37.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

37.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mariadb-sink.kamelet.yaml

第 38 章 掩码字段操作

屏蔽传输中消息中带有恒定值的字段

38.1. 配置选项

下表总结了 **mask-field-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
字段 *	字段	用于掩码的以逗号分隔的字段列表	字符串		
替换 *	替换	替换要屏蔽的字段	字符串		

注意

带有星号(*)标记的字段为必填。

38.2. 依赖项

在运行时，**mask-field-action** Kamelet 依赖于以下依赖项：

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

38.3. 使用方法

本节论述了如何使用 **mask-field-action**。

38.3.1. Knative 操作

您可以使用 **mask-field-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

38.3.1.2. 使用集群 CLI 的步骤

1. 将 **mask-field-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

38.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step mask-field-action -p "step-0.fields=The Fields" -p
"step-0.replacement=The Replacement" channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

38.3.2. Kafka 操作

您可以使用 **mask-field-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

38.3.2.2. 使用集群 CLI 的步骤

1. 将 **mask-field-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

38.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

38.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mask-field-action.kamelet.yaml

第 39 章 MESSAGE TIMESTAMP ROUTER ACTION

更新 topic 字段作为原始主题名称和记录的时间戳字段的功能。

39.1. 配置选项

下表总结了可用于 **message-timestamp-router-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
timestampKeys *	时间戳键	Timestamp 键的逗号分隔列表。时间戳取自第一个找到字段。	字符串		
timestampFormat	时间戳格式	为与 java.text.SimpleDateFormat 的时间戳格式字符串。	字符串	**"yyyyMMdd"**	
timestampKeyFormat	时间戳密钥格式	时间戳密钥的格式。可能的值有 'timestamp' 或任何与 java.text.SimpleDateFormat 兼容的时间戳的格式字符串。如果是 'timestamp'，该字段将评估为自 1970 年以来的毫秒，比如 UNIX Timestamp。	字符串	**"timestamp"**	
topicFormat	主题格式	分别包含 '$[topic]' 和 '$[timestamp]' 的字符串，作为主题和时间戳的占位符。	字符串	**"topic-$[timestamp]"**	

注意

带有星号(*)标记的字段为必填。

39.2. 依赖项

在运行时，**message-timestamp-router-action** Kamelet 依赖于以下依赖项：

- mvn:org.apache.camel.kamelets:camel-kamelets-utils:1.0.0.fuse-800048-redhat-00001
- Camel:jackson
- camel:kamelet
- Camel:core

39.3. 使用方法

本节论述了如何使用 **message-timestamp-router-action**。

39.3.1. Knative 操作

您可以使用 **message-timestamp-router-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

39.3.1.2. 使用集群 CLI 的步骤

1. 将 **message-timestamp-router-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

```
oc apply -f message-timestamp-router-action-binding.yaml
```

39.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step message-timestamp-router-action -p "step-0.timestampKeys=The Timestamp Keys" channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

39.3.2. Kafka 操作

您可以使用 **message-timestamp-router-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

39.3.2.2. 使用集群 CLI 的步骤

1. 将 **message-timestamp-router-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

```
oc apply -f message-timestamp-router-action-binding.yaml
```

39.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

39.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//message-timestamp-router-action.kamelet.yaml

第 40 章 MONGODB SINK

将文档发送到 MongoDB。

这个 Kamelet 预期 JSON 作为正文。

您可以将属性设置为标头：

- **db-upsert / ce-dbupsert**: 如果数据库应该创建元素（如果该元素不存在）。布尔值

40.1. 配置选项

下表总结了可用于 **mongodb-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
集合 *	MongoDB Collection	设置要绑定到此端点的 MongoDB 集合的名称。	字符串		
数据库 *	MongoDB 数据库	将 MongoDB 数据库的名称设置为目标。	字符串		
主机 *	MongoDB 主机	host:port 格式的 MongoDB 主机地址的逗号分隔列表。	字符串		
createCollection	集合	如果初始创建集合不存在，则进行创建。	布尔值	**false**	
password	MongoDB 密码	用于访问 MongoDB 的用户密码。	字符串		
username	MongoDB Username	用于访问 MongoDB 的用户名.	字符串		
writeConcern	写入 Concern	为写入操作配置 MongoDB 请求的速度，可能的值是 ACKNOWLEDGED、W1、W2、W3、UNACKNOWLEDGED、JOURNALED、MAJORITY。	字符串		

注意

带有星号(*)标记的字段为必填。

40.2. 依赖项

运行时，**mongodb-sink** Kamelet 依赖于以下依赖项：

- camel:kamelet
- camel:mongodb
- Camel:jackson

40.3. 使用方法

本节论述了如何使用 **mongodb-sink**。

40.3.1. Knative Sink

您可以将 **mongodb-sink** Kamelet 用作 Knative sink，方法是将其绑定到 Knative 对象。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

40.3.1.2. 使用集群 CLI 的步骤

1. 将 **mongodb-sink-binding.yaml** 文件保存到本地驱动器，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

40.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

-

```
kamel bind channel:mychannel mongodb-sink -p "sink.collection=The MongoDB Collection" -p
"sink.database=The MongoDB Database" -p "sink.hosts=The MongoDB Hosts"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

40.3.2. Kafka Sink

您可以将 **mongodb-sink** Kamelet 用作 Kafka 接收器，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

40.3.2.2. 使用集群 CLI 的步骤

1. 将 **mongodb-sink-binding.yaml** 文件保存到本地驱动器，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

40.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

40.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mongodb-sink.kamelet.yaml

第 41 章 MONGODB 源

使用 MongoDB 文档。

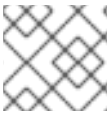
如果启用了 persistentTailTracking 选项，使用者将跟踪上次使用的消息，然后在下一次重新启动时，消耗来自该消息。如果启用 persistentTailTracking，则必须提供 tailTrackIncreasingField（默认为可选）。

如果没有启用 persistentTailTracking 选项，消费者会消耗整个集合，并等待新的文档供使用。

41.1. 配置选项

下表总结了 **mongodb-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
集合 *	MongoDB Collection	设置要绑定到此端点的 MongoDB 集合的名称。	字符串		
数据库 *	MongoDB 数据库	将 MongoDB 数据库的名称设置为目标。	字符串		
主机 *	MongoDB 主机	host:port 格式的 MongoDB 主机地址的逗号分隔列表。	字符串		
密码 *	MongoDB 密码	用于访问 MongoDB 的用户密码。	字符串		
用户名 *	MongoDB Username	用于访问 MongoDB 的用户名.该用户名必须存在于 MongoDB 的身份验证数据库中 (authenticationDatabase)。默认情况下，MongoDB authenticationDatabase 为 'admin'。	字符串		
persistentTailTracking	MongoDB 持久跟踪	启用持久跟踪跟踪，这是在系统重启后跟踪最后一次使用的消息的机制。下次系统启动后，端点将从其上停止的滑动记录中恢复光标。	布尔值	**false**	
tailTrackIncreasingField	MongoDB Tail Track Increasing 字段	传入 记录中的关联字段（即增加的性质），每次生成时都会用来定位尾部光标。	字符串		



注意

带有星号(*)标记的字段为必填。

41.2. 依赖项

mongodb-source Kamelet 运行时依赖于以下依赖项：

- camel:kamelet
- camel:mongodb
- Camel:jackson

41.3. 使用方法

本节论述了如何使用 **mongodb-source**。

41.3.1. Knative Source

您可以将 **mongodb-source** Kamelet 用作 Knative 源，方法是将其绑定到 Knative 对象。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

41.3.1.2. 使用集群 CLI 的步骤

1. 将 **mongodb-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。

2. 使用以下命令运行源：

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

41.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

41.3.2. Kafka Source

您可以将 **mongodb-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

41.3.2.2. 使用集群 CLI 的步骤

1. 将 **mongodb-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。

2. 使用以下命令运行源：

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

41.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

41.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mongodb-source.kamelet.yaml

第 42 章 MYSQL SINK

发送数据到 MySQL 数据库。

这个 Kamelet 预期 JSON 作为正文。JSON 字段和参数之间的映射由键进行，因此如果您有以下查询：

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

Kamelet 需要作为输入内容接收，如下所示：

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

42.1. 配置选项

下表总结了 **mysql-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
databaseName *	数据库名称	我们指向的数据库名称	字符串		
密码 *	密码	用于访问安全 MySQL 数据库的密码	字符串		
查询 *	查询	要针对 MySQL 数据库执行的查询	字符串		**"INSERT INTO 帐户 (username,city)VALUES(:#username,:#city)"**
serverName *	服务器名称	数据源的服务器名称	字符串		**"localhost"**
用户名 *	用户名	用于访问安全 MySQL 数据库的用户名	字符串		
serverPort	服务器端口	数据源的服务器端口	字符串	**3306**	

注意

带有星号(*)标记的字段为必填。

42.2. 依赖项

在运行时，**mysql-sink** Kamelet 依赖于以下依赖项：

- Camel:jackson
- camel:kamelet

- camel:sql
- mvn:org.apache.commons:commons-dbcp2:2.7.0.redhat-00001
- mvn:mysql:mysql-connector-java

42.3. 使用方法

本节论述了如何使用 **mysql-sink**。

42.3.1. Knative Sink

您可以将 **mysql-sink** Kamelet 用作 Knative 接收器，方法是将其绑定到 Knative 对象。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

42.3.1.2. 使用集群 CLI 的步骤

1. 将 **mysql-sink-binding.yaml** 文件保存到本地驱动器，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

42.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

42.3.2. Kafka Sink

您可以将 **mysql-sink** Kamelet 用作 Kafka 接收器，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

42.3.2.2. 使用集群 CLI 的步骤

1. 将 **mysql-sink-binding.yaml** 文件保存到本地驱动器，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

42.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

42.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//mysql-sink.kamelet.yaml

第 43 章 POSTGRESQL SINK

将数据发送到 PostgreSQL 数据库。

这个 Kamelet 预期 JSON 作为正文。JSON 字段和参数之间的映射由键进行，因此如果您有以下查询：

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

Kamelet 需要作为输入内容接收，如下所示：

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

43.1. 配置选项

下表总结了 **postgresql-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
databaseName *	数据库名称	我们指向的数据库名称	字符串		
密码 *	密码	用于访问受保护的 PostgreSQL 数据库的密码	字符串		
查询 *	查询	要对 PostgreSQL 数据库执行的查询	字符串		**"INSERT INTO 帐户 (username,city)VALUES(:#username,:#city)"**
serverName *	服务器名称	数据源的服务器名称	字符串		**"localhost"**
用户名 *	用户名	用于访问受保护的 PostgreSQL 数据库的用户名	字符串		
serverPort	服务器端口	数据源的服务器端口	字符串	**5432**	

注意

带有星号(*)标记的字段为必填。

43.2. 依赖项

在运行时，**postgresql-sink** Kamelet 依赖于以下依赖项：

- Camel:jackson
- camel:kamelet

- camel:sql
- mvn:org.postgresql:postgresql
- mvn:org.apache.commons:commons-dbcp2:2.7.0.redhat-00001

43.3. 使用方法

本节论述了如何使用 **postgresql-sink**。

43.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **postgresql-sink** Kamelet 作为 Knative sink。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

43.3.1.2. 使用集群 CLI 的步骤

1. 将 **postgresql-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

43.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

43.3.2. Kafka Sink

您可以将 **postgresql-sink** Kamelet 用作 Kafka 接收器，方法是将其绑定到 Kafka 主题。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

43.3.2.2. 使用集群 CLI 的步骤

1. 将 **postgresql-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

43.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

43.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//postgresql-sink.kamelet.yaml

第 44 章 PREDICATE 过滤器操作

基于 JsonPath 表达式进行过滤

44.1. 配置选项

下表总结了 **predicate-filter-action** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
表达式 *	表达式	JsonPath 表达式，用于评估外部圆括号。由于这是一个过滤器，因此表达式为 negation，这表示，如果示例中的 foo 字段等于 John，则消息将提前过滤，否则将过滤掉。	字符串		**"@.foo =~ /.*John/"**

注意

带有星号(*)标记的字段为必填。

44.2. 依赖项

在运行时，**predicate-filter-action** Kamelet 依赖于以下依赖项：

- Camel:core
- camel:kamelet
- camel:jsonpath

44.3. 使用方法

这部分论述了如何使用 **predicate-filter-action**。

44.3.1. Knative 操作

您可以使用 **predicate-filter-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

44.3.1.2. 使用集群 CLI 的步骤

1. 将 **predicate-filter-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f predicate-filter-action-binding.yaml
```

44.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step predicate-filter-action -p "step-0.expression=@.foo
=~ /.*John/" channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

44.3.2. Kafka 操作

您可以使用 **predicate-filter-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

44.3.2.2. 使用集群 CLI 的步骤

1. 将 **predicate-filter-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f predicate-filter-action-binding.yaml
```

44.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step predicate-filter-action -p "step-0.expression=@.foo =~ /.*John/" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

44.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//predicate-filter-action.kamelet.yaml

第 45 章 PROTOBUF DESERIALIZE ACTION

对 Protobuf 进行序列化有效负载

45.1. 配置选项

下表总结了 **protobuf-deserialize-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
schema *	模式	在序列化（作为单行）中使用的 Protobuf 模式	字符串		**"message Person { required string first = 1; required string last = 2; }"**

注意

带有星号(*)标记的字段为必填。

45.2. 依赖项

在运行时，**protobuf-deserialize-action** Kamelet 依赖于以下依赖项：

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

45.3. 使用方法

本节论述了如何使用 **protobuf-deserialize-action**。

45.3.1. Knative 操作

您可以使用 **protobuf-deserialize-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

45.3.1.2. 使用集群 CLI 的步骤

1. 将 **protobuf-deserialize-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

45.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

45.3.2. Kafka 操作

您可以使用 **protobuf-deserialize-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

45.3.2.2. 使用集群 CLI 的步骤

1. 将 **protobuf-deserialize-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

45.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

45.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//protobuf-deserialize-action.kamelet.yaml

第 46 章 PROTOBUF SERIALIZE ACTION

为 Protobuf 序列化有效负载

46.1. 配置选项

下表总结了 **protobuf-serialize-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
schema *	模式	在序列化（作为单行）中使用的 Protobuf 模式	字符串		**"message Person { required string first = 1; required string last = 2; }"**

注意

带有星号(*)标记的字段为必填。

46.2. 依赖项

在运行时，**protobuf-serialize-action** Kamelet 依赖于以下依赖项：

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

46.3. 使用方法

本节论述了如何使用 **protobuf-serialize-action**。

46.3.1. Knative 操作

您可以使用 **protobuf-serialize-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

46.3.1.2. 使用集群 CLI 的步骤

1. 将 **protobuf-serialize-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

46.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

46.3.2. Kafka 操作

您可以使用 **protobuf-serialize-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

46.3.2.2. 使用集群 CLI 的步骤

1. 将 **protobuf-serialize-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要编辑它。
2. 使用以下命令运行操作：

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

46.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

46.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//protobuf-serialize-action.kamelet.yaml

第 47 章 正则表达式路由器操作

使用配置的正则表达式和替换字符串更新目的地

47.1. 配置选项

下表总结了针对 **regex-router-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
regex *	正则表达式	正则表达式用于目的地	字符串		
替换 *	替换	匹配时替换	字符串		

注意

带有星号(*)标记的字段为必填。

47.2. 依赖项

在运行时，**regex-router-action** Kamelet 依赖于以下依赖项：

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

47.3. 使用方法

本节论述了如何使用 **regex-router-action**。

47.3.1. Knative 操作

您可以使用 **regex-router-action** Kamelet 作为 Knative 绑定中的中间步骤。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

47.3.1.2. 使用集群 CLI 的步骤

1. 将 **regex-router-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f regex-router-action-binding.yaml
```

47.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step regex-router-action -p "step-0.regex=The Regex" -p
"step-0.replacement=The Replacement" channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

47.3.2. Kafka 操作

您可以使用 **regex-router-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

47.3.2.2. 使用集群 CLI 的步骤

1. 将 **regex-router-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f regex-router-action-binding.yaml
```

47.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

47.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//regex-router-action.kamelet.yaml

第 48 章 替换字段操作

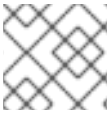
使用传输中的消息中的不同密钥替换 field。

- 所需参数 'renames' 是一个用逗号分开的冒号分隔列表，如 'foo:bar,abc:xyz'，它代表字段重命名映射。
- 可选参数"enabled"表示要包含的字段。如果指定，则生成的消息中只会包含 named 字段。
- 可选参数 'disabled' 代表要排除的字段。如果指定，列出的字段将不包括在结果消息中。这优先于 'enabled' 参数。
- 'enabled' 参数的默认值为 'all'，因此将包括有效负载的所有字段。
- 'disabled' 参数的默认值为 'none'，因此没有有效负载的字段将排除。

48.1. 配置选项

下表总结了用于 **replace-field-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
重命名 *	renames	使用要重命名的新值的以逗号分隔的字段列表	字符串		**"foo:bar,c1:c2"**
Disabled	Disabled	要禁用以逗号分隔的字段列表	字符串	"无"	
enabled	Enabled	启用以逗号分隔的字段列表	字符串	"所有"	



注意

带有星号(*)标记的字段为必填。

48.2. 依赖项

在运行时，**replace-field-action** Kamelet 依赖于以下依赖项：

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

48.3. 使用方法

本节论述了如何使用 **replace-field-action**。

48.3.1. Knative 操作

您可以使用 **replace-field-action** Kamelet 作为 Knative 绑定中的中间步骤。

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:
      renames: "foo:bar,c1:c2"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

48.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

48.3.1.2. 使用集群 CLI 的步骤

1. 将 **replace-field-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

48.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step replace-field-action -p "step-
0.renames=foo:bar,c1:c2" channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

48.3.2. Kafka 操作

您可以使用 **replace-field-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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:
      renames: "foo:bar,c1:c2"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

48.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

48.3.2.2. 使用集群 CLI 的步骤

1. 将 **replace-field-action-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

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

48.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step replace-field-action -p "step-
0.renames=foo:bar,c1:c2" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

48.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//replace-field-action.kamelet.yaml

第 49 章 SALESFORCE 源

从 Salesforce 网站接收更新。

49.1. 配置选项

下表总结了 Mixer **-source** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
clientID *	消费者密钥	Salesforce 应用程序消费者密钥	字符串		
clientSecret *	消费者机密	Salesforce 应用程序消费者 secret	字符串		
密码 *	密码	Salesforce 用户密码	字符串		
查询 *	查询	在 Salesforce 上执行的查询	字符串		**"SELECT Id, Name, Email, Phone FROM Contact"**
topicName *	主题名称	要使用的主题/频道的名称	字符串		**"ContactTopic"**
userName *	用户名	Salesforce 用户名	字符串		
loginUrl	登录 URL	Salesforce 实例登录 URL	字符串	**"https://login.salesforce.com"**	

注意

带有星号(*)标记的字段为必填。

49.2. 依赖项

在运行时，round **force-source** Kamelet 依赖于以下依赖项：

- Camel:jackson
- camel:salesforce
- mvn:org.apache.camel.k:camel-k-kamelet-reify
- camel:kamelet

49.3. 使用方法

本节介绍了如何使用 **salesforce-source**。

49.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，将 Clair **-source** Kamelet 用作 Knative 源。

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. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

49.3.1.2. 使用集群 CLI 的步骤

1. 将 **dashboard-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

49.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

49.3.2. Kafka Source

您可以通过将其绑定到 Kafka 主题，将roundforce **-source** Kamelet 用作 Kafka 源。

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. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

49.3.2.2. 使用集群 CLI 的步骤

1. 将 **dashboard-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

49.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

49.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//salesforce-source.kamelet.yaml

第 50 章 SALESFORCE CREATE SINK

在 Salesforce 中创建对象。邮件正文必须包含 salesforce 对象的 JSON。

示例正文：{ "Phone": "555", "Name": "Antonia", "LastName": "Garcia" }

50.1. 配置选项

下表总结了 Mixer **-create-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
clientID *	消费者密钥	Salesforce 应用程序消费者密钥	字符串		
clientSecret *	消费者机密	Salesforce 应用程序消费者 secret	字符串		
密码 *	密码	Salesforce 用户密码	字符串		
userName *	用户名	Salesforce 用户名	字符串		
loginUrl	登录 URL	Salesforce 实例登录 URL	字符串	**"https://login.salesforce.com"**	
sObjectName	对象名称	对象的类型	字符串		**"联系人"**

注意

带有星号(*)标记的字段为必填。

50.2. 依赖项

在运行时，round **force-create-sink** Kamelet 依赖于以下依赖项：

- camel:salesforce
- camel:kamelet

50.3. 使用方法

这部分论述了如何使用 **salesforce-create-sink**。

50.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **salesforce-create-sink** Kamelet 作为 Knative sink。

salesforce-create-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-create-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-create-sink
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      userName: "The Username"
```

50.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

50.3.1.2. 使用集群 CLI 的步骤

1. 将 **dashboard-create-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

```
oc apply -f salesforce-create-sink-binding.yaml
```

50.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
kamel bind channel:mychannel salesforce-create-sink -p "sink.clientId=The Consumer Key" -p
"sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.userName=The
Username"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

50.3.2. Kafka Sink

您可以通过将其绑定到 Kafka 主题，使用 **salesforce-create-sink** Kamelet 作为 Kafka sink。

salesforce-create-sink-binding.yaml

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

```
  name: salesforce-create-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: salesforce-create-sink
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      userName: "The Username"
```

50.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

50.3.2.2. 使用集群 CLI 的步骤

1. 将 **dashboard-create-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

```
oc apply -f salesforce-create-sink-binding.yaml
```

50.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic salesforce-create-sink -p "sink.clientId=The Consumer Key" -p "sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.userName=The Username"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

50.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//salesforce-create-sink.kamelet.yaml

第 51 章 SALESFORCE DELETE SINK

从 Salesforce 中删除对象。收到的正文必须是含有两个键的 JSON：sObjectId 和 sObjectName。

示例正文：{ "sObjectId": "XXXXX0", "sObjectName": "Contact" }

51.1. 配置选项

下表总结了在 Mixer **-delete-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
clientID *	消费者密钥	Salesforce 应用程序消费者密钥	字符串		
clientSecret *	消费者机密	Salesforce 应用程序消费者 secret	字符串		
密码 *	密码	Salesforce 用户密码	字符串		
userName *	用户名	Salesforce 用户名	字符串		
loginUrl	登录 URL	Salesforce 实例登录 URL	字符串	**"https://login.salesforce.com"**	

注意

带有星号(*)标记的字段为必填。

51.2. 依赖项

在运行时，round **force-delete-sink** Kamelet 依赖于以下依赖项：

- camel:salesforce
- camel:kamelet
- Camel:core
- camel:jsonpath

51.3. 使用方法

本节介绍了如何使用 **salesforce-delete-sink**。

51.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **53-delete-** sink Kamelet 作为 Knative sink。

salesforce-delete-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-delete-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-delete-sink
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      userName: "The Username"
```

51.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

51.3.1.2. 使用集群 CLI 的步骤

1. 将 **dashboard-delete-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

```
oc apply -f salesforce-delete-sink-binding.yaml
```

51.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
kamel bind channel:mychannel salesforce-delete-sink -p "sink.clientId=The Consumer Key" -p
"sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.userName=The
Username"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

51.3.2. Kafka Sink

您可以通过将其绑定到 Kafka 主题，使用 **53-delete-** sink Kamelet 作为 Kafka sink。

salesforce-delete-sink-binding.yaml

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

```
  name: salesforce-delete-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: salesforce-delete-sink
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      userName: "The Username"
```

51.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

51.3.2.2. 使用集群 CLI 的步骤

1. 将 **dashboard-delete-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

```
oc apply -f salesforce-delete-sink-binding.yaml
```

51.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic salesforce-delete-sink -p "sink.clientId=The Consumer Key" -p "sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.userName=The Username"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

51.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//salesforce-delete-sink.kamelet.yaml

第 52 章 SALESFORCE UPDATE SINK

更新 Salesforce 中的对象。收到的正文必须包含每个属性的 JSON 键值对，才能作为参数提供更新和 sObjectName 和 sObjectId。

键值对示例： { "Phone": "1234567890", "Name": "Antonia" }

52.1. 配置选项

下表总结了 Mixer **-update-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
clientID *	消费者密钥	Salesforce 应用程序消费者密钥	字符串		
clientSecret *	消费者机密	Salesforce 应用程序消费者 secret	字符串		
密码 *	密码	Salesforce 用户密码	字符串		
sObjectId *	对象 Id	对象的 ID。仅在使用键值对时才需要。	字符串		
sObjectName *	对象名称	对象的类型。仅在使用键值对时才需要。	字符串		**"联系人"**
userName *	用户名	Salesforce 用户名	字符串		
loginUrl	登录 URL	Salesforce 实例登录 URL	字符串	**"https://login.salesforce.com"**	

注意

带有星号(*)标记的字段为必填。

52.2. 依赖项

在运行时，nameserver **-update-sink** Kamelet 依赖于以下依赖项：

- camel:salesforce
- camel:kamelet

52.3. 使用方法

这部分论述了如何使用 **salesforce-update-sink**。

52.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **salesforce-update-sink** Kamelet 作为 Knative sink。

salesforce-update-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-update-sink-binding
spec:
  source:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: salesforce-update-sink
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      sObjectId: "The Object Id"
      sObjectName: "Contact"
      userName: "The Username"
```

52.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

52.3.1.2. 使用集群 CLI 的步骤

1. 将 **dashboard-update-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

```
oc apply -f salesforce-update-sink-binding.yaml
```

52.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
kamel bind channel:mychannel salesforce-update-sink -p "sink.clientId=The Consumer Key" -p "sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.sObjectId=The Object Id" -p "sink.sObjectName=Contact" -p "sink.userName=The Username"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

52.3.2. Kafka Sink

您可以通过将其绑定到 Kafka 主题，使用 **salesforce-update-sink** Kamelet 作为 Kafka sink。

salesforce-update-sink-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: salesforce-update-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: salesforce-update-sink
    properties:
      clientId: "The Consumer Key"
      clientSecret: "The Consumer Secret"
      password: "The Password"
      sObjectId: "The Object Id"
      sObjectName: "Contact"
      userName: "The Username"
```

52.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

52.3.2.2. 使用集群 CLI 的步骤

1. 将 **dashboard-update-sink-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

```
oc apply -f salesforce-update-sink-binding.yaml
```

52.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic salesforce-update-sink -p "sink.clientId=The Consumer Key" -p "sink.clientSecret=The Consumer Secret" -p "sink.password=The Password" -p "sink.sObjectId=The Object Id" -p "sink.sObjectName=Contact" -p "sink.userName=The Username"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

52.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//salesforce-update-sink.kamelet.yaml

第 53 章 SFTP SINK

将数据发送到 SFTP 服务器。

Kamelet 期望设置以下标头：

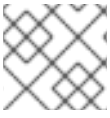
- **文件 / ce-file**: 作为要上传的文件名

如果标头没有设置交换 ID，则将用作文件名。

53.1. 配置选项

下表总结了 **sftp-sink** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
connectionHost *	连接主机	FTP 服务器的主机名	字符串		
connectionPort *	连接端口	FTP 服务器的端口	字符串	**22**	
directoryName *	目录名称	起始目录	字符串		
密码 *	密码	用于访问 FTP 服务器的密码	字符串		
用户名 *	用户名	用于访问 FTP 服务器的用户名	字符串		
fileExist	文件保护	如果文件已存在，如何的行为。有 4 个枚举值，值可以是 Override、Append、Fail 或 Ignore 之一。	字符串	**"Override"**	
passiveMode	被动模式	设置被动模式连接	布尔值	**false**	



注意

带有星号(*)标记的字段为必填。

53.2. 依赖项

在运行时，**sftp-sink** Kamelet 依赖于以下依赖项：

- camel:ftp
- Camel:core

- camel:kamelet

53.3. 使用方法

本节介绍了如何使用 **sftp-sink**。

53.3.1. Knative Sink

您可以通过将其绑定到 Knative 对象，使用 **sftp-sink** Kamelet 作为 Knative sink。

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"
```

53.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

53.3.1.2. 使用集群 CLI 的步骤

1. 将 **sftp-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

53.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

53.3.2. Kafka Sink

您可以将 **sftp-sink** Kamelet 用作 Kafka 接收器，方法是将其绑定到 Kafka 主题。

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"
```

53.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

53.3.2.2. 使用集群 CLI 的步骤

1. 将 **sftp-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

53.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

53.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//sftp-sink.kamelet.yaml

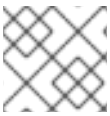
第 54 章 SFTP 源

从 SFTP 服务器接收数据。

54.1. 配置选项

下表总结了 **sftp-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
connectionHost *	连接主机	SFTP 服务器的主机名	字符串		
connectionPort *	连接端口	FTP 服务器的端口	字符串	**22**	
directoryName *	目录名称	起始目录	字符串		
密码 *	密码	访问 SFTP 服务器的密码	字符串		
用户名 *	用户名	用于访问 SFTP 服务器的用户名	字符串		
idempotent	幂等性	跳过已处理的文件。	布尔值	**true**	
passiveMode	被动模式	设置被动模式连接	布尔值	**false**	
递归	递归	如果某个目录，也会查找所有子目录中的文件。	布尔值	**false**	



注意

带有星号(*)标记的字段为必填。

54.2. 依赖项

在运行时，**sftp-source** Kamelet 依赖于以下依赖项：

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

54.3. 使用方法

这部分论述了如何使用 **sftp-source**。

54.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，将 **sftp-source** Kamelet 用作 Knative 源。

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
```

54.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

54.3.1.2. 使用集群 CLI 的步骤

1. 将 **sftp-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

54.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

54.3.2. Kafka Source

您可以将 **sftp-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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
```

54.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

54.3.2.2. 使用集群 CLI 的步骤

1. 将 **sftp-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

54.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
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
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

54.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//sftp-source.kamelet.yaml

第 55 章 SLACK 源

从 Slack 频道接收消息。

55.1. 配置选项

下表总结了 **slack-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
频道 *	Channel	用于接收信息的 Slack 频道	字符串		**"#myroom"**
令牌 *	令牌	用于访问 Slack 的令牌。需要 Slack app。此应用程序需要具有 channel:history 和 channels:read 权限。Bot User OAuth Access Token 是所需的令牌类型。	字符串		

注意

带有星号(*)标记的字段为必填。

55.2. 依赖项

在运行时，**slack-source** Kamelet 依赖于以下依赖项：

- camel:kamelet
- camel:slack
- Camel:jackson

55.3. 使用方法

这部分论述了如何使用 **slack-source**。

55.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，将 **slack-source** Kamelet 用作 Knative 源。

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
```

55.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

55.3.1.2. 使用集群 CLI 的步骤

1. 将 **slack-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

55.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
kamel bind slack-source -p "source.channel=#myroom" -p "source.token=The Token"
channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

55.3.2. Kafka Source

您可以将 **slack-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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
```

55.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

55.3.2.2. 使用集群 CLI 的步骤

1. 将 **slack-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

55.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
kamel bind slack-source -p "source.channel=#myroom" -p "source.token=The Token"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

55.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//slack-source.kamelet.yaml

第 56 章 MICROSOFT SQL SERVER SINK

将数据发送到 Microsoft SQL Server 数据库。

这个 Kamelet 预期 JSON 作为正文。JSON 字段和参数之间的映射由键进行，因此如果您有以下查询：

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

Kamelet 需要作为输入内容接收，如下所示：

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

56.1. 配置选项

下表总结了 **sqlserver-sink** Kamelet 可用的配置选项：

属性	名称	描述	类型	默认	示例
databaseName *	数据库名称	我们指向的数据库名称	字符串		
密码 *	密码	用于访问安全 SQL Server 数据库的密码	字符串		
查询 *	查询	要针对 SQL Server 数据库执行的 Query	字符串		**"INSERT INTO 帐户 (username,city)VALUES(:#username,:#city)"**
serverName *	服务器名称	数据源的服务器名称	字符串		**"localhost"**
用户名 *	用户名	用于访问安全 SQL Server 数据库的用户名	字符串		
serverPort	服务器端口	数据源的服务器端口	字符串	**1433**	

注意

带有星号(*)标记的字段为必填。

56.2. 依赖项

在运行时，**sqlserver-sink** Kamelet 依赖于以下依赖项：

- 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

56.3. 使用方法

本节论述了如何使用 **sqlserver-sink**。

56.3.1. Knative Sink

您可以将 **sqlserver-sink** Kamelet 用作 Knative sink，方法是将其绑定到 Knative 对象。

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"
```

56.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

56.3.1.2. 使用集群 CLI 的步骤

1. 将 **sqlserver-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

56.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

56.3.2. Kafka Sink

您可以将 **sqlserver-sink** Kamelet 用作 Kafka sink，方法是将其绑定到 Kafka 主题。

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"
```

56.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

56.3.2.2. 使用集群 CLI 的步骤

1. 将 **sqlserver-sink-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行 sink：

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

56.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行 sink：

```
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"
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

56.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//sqlserver-sink.kamelet.yaml

第 57 章 TELEGRAM 源

接收人们发送给您的 Telegram bot 的所有信息。

要创建 bot，使用 Telegram 应用联系 @botfather 帐户。

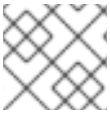
源将以下标头附加到消息：

- **chat-id** / **ce-chatid** ：消息的聊天 ID

57.1. 配置选项

下表总结了可用于 **telegram-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
authorizationToken *	令牌	访问 Telegram 上的 bot 的令牌。您可以从 Telegram @botfather 获得它。	字符串		



注意

带有星号(*)标记的字段为必填。

57.2. 依赖项

在运行时，**telegram-source** Kamelet 依赖于以下依赖项：

- Camel:jackson
- camel:kamelet
- Camel:telegram
- Camel:core

57.3. 使用方法

这部分论述了如何使用 **telegram-source**。

57.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，使用 **telegram-source** Kamelet 作为 Knative 源。

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
```

57.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

57.3.1.2. 使用集群 CLI 的步骤

1. 将 **telegram-source-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

57.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
kamel bind telegram-source -p "source.authorizationToken=The Token" channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

57.3.2. Kafka Source

您可以通过将其绑定到 Kafka 主题，使用 **telegram-source** Kamelet 作为 Kafka 源。

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
```

57.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

57.3.2.2. 使用集群 CLI 的步骤

1. 将 **telegram-source-binding.yaml** 文件保存到您的本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

57.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
kamel bind telegram-source -p "source.authorizationToken=The Token"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

57.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//telegram-source.kamelet.yaml

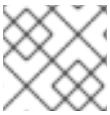
第 58 章 THROTTLE ACTION

Throttle 操作可让您确保特定的 sink 不会被加载。

58.1. 配置选项

下表总结了可用于 **throttle-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
信息 *	消息号	要在时间段集中发送的消息数量	整数		**10**
timePeriod	时间时间	设置最大请求数有效的时间段（以毫秒为单位）	字符串	**"1000"**	



注意

带有星号(*)标记的字段为必填。

58.2. 依赖项

在运行时，**throttle-action** Kamelet 依赖于以下依赖项：

- Camel:core
- camel:kamelet

58.3. 使用方法

本节论述了如何使用 **throttle-action**。

58.3.1. Knative 操作

您可以使用 **throttle-action** Kamelet 作为 Knative 绑定中的中间步骤。

throttle-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: throttle-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: throttle-action
    properties:
      messages: 1
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

58.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

58.3.1.2. 使用集群 CLI 的步骤

1. 将 **throttle-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f throttle-action-binding.yaml
```

58.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step throttle-action -p "step-0.messages=10"
channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

58.3.2. Kafka 操作

您可以使用 **throttle-action** Kamelet 作为 Kafka 绑定中的中间步骤。

throttle-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: throttle-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: throttle-action
  properties:
    messages: 1
sink:
  ref:
    kind: KafkaTopic
    apiVersion: kafka.strimzi.io/v1beta1
    name: my-topic
```

58.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

58.3.2.2. 使用集群 CLI 的步骤

1. 将 **throttle-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f throttle-action-binding.yaml
```

58.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step throttle-action -p "step-0.messages=1"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

58.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//throttle-action.kamelet.yaml

第 59 章 计时器源

生成带有自定义有效负载的定期事件。

59.1. 配置选项

下表总结了 **timer-source** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
Message *	消息	要生成的消息	字符串		**"hello world"**
contentType	内容类型	正在生成的消息的内容类型	字符串	**"text/plain"**	
周期	时期	两个事件之间的间隔，以毫秒为单位	整数	**1000**	

注意

带有星号(*)标记的字段为必填。

59.2. 依赖项

运行时，**timer-source** Kamelet 依赖于以下依赖项：

- Camel:core
- Camel:timer
- camel:kamelet

59.3. 使用方法

本节论述了如何使用 **timer-source**。

59.3.1. Knative Source

您可以通过将其绑定到 Knative 对象，将 **timer-source** Kamelet 用作 Knative 源。

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
```

59.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

59.3.1.2. 使用集群 CLI 的步骤

1. 将 **timer-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

59.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
kamel bind timer-source -p "source.message=hello world" channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

59.3.2. Kafka Source

您可以将 **timer-source** Kamelet 用作 Kafka 源，方法是将其绑定到 Kafka 主题。

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
```

59.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

59.3.2.2. 使用集群 CLI 的步骤

1. 将 **timer-source-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行源：

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

59.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行源：

```
kamel bind timer-source -p "source.message=hello world" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

59.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//timer-source.kamelet.yaml

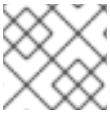
第 60 章 时间戳路由器操作

更新 topic 字段作为原始主题名称和记录时间戳的功能。

60.1. 配置选项

下表总结了可用于 **timestamp-router-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
timestampFormat	时间戳格式	为与 java.text.SimpleDateFormat 的时间戳格式字符串。	字符串	**"yyyyMMdd"**	
timestampHeaderName	时间戳标头名称	包含时间戳的标头的名称	字符串	**"kafka.TIMESTAMP"**	
topicFormat	主题格式	分别包含 '$[topic]' 和 '$[timestamp]' 的字符串，作为主题和时间戳的占位符。	字符串	**"topic-$[timestamp]"**	



注意

带有星号(*)标记的字段为必填。

60.2. 依赖项

在运行时，**timestamp-router-action** Kamelet 依赖于以下依赖项：

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

60.3. 使用方法

本节论述了如何使用 **timestamp-router-action**。

60.3.1. Knative 操作

您可以使用 **timestamp-router-action** Kamelet 作为 Knative 绑定中的中间步骤。

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
```

60.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

60.3.1.2. 使用集群 CLI 的步骤

1. 将 **timestamp-router-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f timestamp-router-action-binding.yaml
```

60.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step timestamp-router-action channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

60.3.2. Kafka 操作

您可以使用 **timestamp-router-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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
```

60.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

60.3.2.2. 使用集群 CLI 的步骤

1. 将 **timestamp-router-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f timestamp-router-action-binding.yaml
```

60.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

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

这个命令会在集群的当前命名空间中创建 KameletBinding。

60.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//timestamp-router-action.kamelet.yaml

第 61 章 KEY ACTION 的值

将 Kafka record 键替换为从正文中字段子集创建新密钥。

61.1. 配置选项

下表总结了可用于 **value-to-key-action** Kamelet 的配置选项：

属性	名称	描述	类型	默认	示例
字段 *	字段	用于组成新键的字段逗号分隔列表	字符串		

注意

带有星号(*)标记的字段为必填。

61.2. 依赖项

在运行时，**value-to-key-action** Kamelet 依赖于以下依赖项：

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

61.3. 使用方法

本节论述了如何使用 **value-to-key-action**。

61.3.1. Knative 操作

您可以使用 **value-to-key-action** Kamelet 作为 Knative 绑定中的中间步骤。

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
```

61.3.1.1. 前提条件

请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

61.3.1.2. 使用集群 CLI 的步骤

1. 将 **value-to-key-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f value-to-key-action-binding.yaml
```

61.3.1.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step value-to-key-action -p "step-0.fields=The Fields"
channel:mychannel
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

61.3.2. Kafka 操作

您可以使用 **value-to-key-action** Kamelet 作为 Kafka 绑定中的中间步骤。

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
```

61.3.2.1. 先决条件

确保已在 OpenShift 集群中安装了 **AMQ Streams** Operator，并在当前命名空间中创建一个名为 **my-topic** 的主题。还请确定您已将 **"Red Hat Integration - Camel K"** 安装到您连接的 OpenShift 集群中。

61.3.2.2. 使用集群 CLI 的步骤

1. 将 **value-to-key-action-binding.yaml** 文件保存到本地驱动器中，然后根据需要为您的配置进行编辑。
2. 使用以下命令运行操作：

```
oc apply -f value-to-key-action-binding.yaml
```

61.3.2.3. 使用 Kamel CLI 的步骤

使用以下命令配置并运行操作：

```
kamel bind timer-source?message=Hello --step value-to-key-action -p "step-0.fields=The Fields"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

这个命令会在集群的当前命名空间中创建 KameletBinding。

61.4. KAMELET 源文件

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//value-to-key-action.kamelet.yaml