Red Hat Integration 2022.Q3

Kamelets リファレンス

Kamelets リファレンス

Last Updated: 2022-11-30

Red Hat Integration 2022.Q3 Kamelets リファレンス

Kamelets リファレンス

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.

法律上の通知

概要

Kamelets は、外部システムに接続するデータパイプライン作成の複雑さを隠す、再利用可能なルートコンポーネントです。

目次

はじめに

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 AWS DYNAMODB SINK

AWS DynamoDB サービスにデータを送信します。送信されたデータは、指定された AWS DynamoDB テーブルのアイテムを挿入/更新/削除します。

アクセスキー/シークレットキーは、AWS DynamoDB サービスに対する基本的な認証方法です。Kamelet は以下のオプション 'useDefaultCredentialsProvider' も提供するため、これらのパラメーターはオプションとなります。

デフォルトの認証情報プロバイダーを使用する場合、AWS DynamoDB クライアントはこのプロバイダーを通して認証情報を読み込み、静的な認証情報を使用しません。このため、この Kamelet では、アクセスキーとシークレットキーを必須パラメーターとしていません。

この Kamelet は、ボディーとして JSON フィールドを想定しています。JSON フィールドとテーブルの属性値とのマッピングはキーで行われるので、以下のような入力があった場合は、

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

Kamelet は、指定された AWS DynamoDB テーブルにアイテムを挿入/更新し、属性 'username' と 'city' をそれぞれ設定します。JSON オブジェクトには、項目を定義する主キー値を含む必要があることに注意してください。

1.1. 設定オプション

次の表は、**aws-ddb-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
region *	AWS リージョン	以下に接続する AWS リージョン	string		**"eu-west-1"**
table *	テーブル	参照する DynamoDB テーブルの名前	string		
accessKey	アクセスキー	AWS から取得したアクセスキー	string		
operation	操作	実行する操作 (PutItem、UpdateItem、DeleteItem のいずれか)。	string	**"PutItem"**	**"PutItem"**
overrideEndpoint	エンドポイントの上書き	エンドポイント URI をオーバーライドする必要性を設定する。このオプションは uriEndpointOverride 設定と併用する必要があります。	boolean	**false**	

プロパティー	名前	説明	型	デフォルト	例
secretKey	シークレットキー	AWS から取得したシークレットキー	string		
uriEndpointOverride	エンドポイント URI の上書き	オーバーライドするエンドポイント URI を設定します。このオプションは overrideEndpoint オプションと組み合わせて使用する必要があります。	string		
useDefaultCredentialsProvider	デフォルトのクレデンシャルプロバイダー	デフォルトのクレデンシャルプロバイダー経由でクレデンシャルをロードすること、または静的クレデンシャルが渡されることを DynamoDB クライアントは想定すべきかどうかを設定します。	boolean	**false**	
writeCapacity	書き込み容量	テーブルにリソースを書き込むために予約するプロビジョニングされたスループット。	integer	**1**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

1.2. DEPENDENCIES

実行時に、**aws-ddb-sink** Kamelet は以下の依存関係の存在に依存します。

- 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

ここでは、**aws-ddb-sink** の使用方法について説明します。

1.3.1. Knative Sink

aws-ddb-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

1.3.1.2. クラスター CLI の使用手順

1. **aws-ddb-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

1.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

1.3.2. Kafka Sink

aws-ddb-sink Kamelet を Kafka トピックにバインドすることで、Knative のシンクとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

1.3.2.2. クラスター CLI の使用手順

1. **aws-ddb-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

1.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

1.4. KAMELET ソースファイル

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

第2章 AVRO デシリアライズアクション

ペイロードを Avro にデシリアライズ

2.1. 設定オプション

次の表は、**avro-deserialize-action**Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
スキーマ *	スキーマ	シリアライゼーション時に使用する Avro スキーマ (JSON 形式を使用)	string		**"{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\": [{\"name\": \"first\", \"type\": \"string\"}, {\"name\": \"last\", \"type\": \"string\"}]}"**
validate	検証	コンテンツがスキーマに対して検証される必要があるかどうかを示します。	boolean	**true**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

2.2. DEPENDENCIES

実行時に、**avro-deserialize-action** Kamelet は以下の依存関係の存在に依存します。

- 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

ここでは、**avro-deserialize-action** の使用方法について説明します。

2.3.1. Knative Action

avro-deserialize-action Kamelet を Knative バインディングの中間ステップとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

2.3.1.2. クラスター CLI の使用手順

1. **avro-deserialize-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

2.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

2.3.2. Kafka Action

avro-deserialize-action Kamelet を Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

2.3.2.2. クラスター CLI の使用手順

1. **avro-deserialize-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

2.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

2.4. KAMELET ソースファイル

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

第3章 AVRO シリアルアクション

ペイロードと Avro へのシリアライズ

3.1. 設定オプション

次の表は、**avro-serialize-action** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
スキーマ *	スキーマ	シリアライゼーション時に使用する Avro スキーマ (JSON 形式を使用)	string		**"{\"type\": \"record\", \"namespace\": \"com.example\", \"name\": \"FullName\", \"fields\": [{\"name\": \"first\", \"type\": \"string\"}, {\"name\": \"last\", \"type\": \"string\"}]}"**
validate	検証	コンテンツがスキーマに対して検証される必要があるかどうかを示します。	boolean	**true**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

3.2. DEPENDENCIES

実行時に、**avro-serialize-action** Kamelet は以下の依存関係の存在に依存しています。

- 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

ここでは、**avro-serialize-action** の使用方法について説明します。

3.3.1. Knative Action

avro-serialize-action Kamelet を Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

3.3.1.2. クラスター CLI の使用手順

1. **avro-serialize-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

3.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

3.3.2. Kafka Action

avro-serialize-action Kamelet を Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

3.3.2.2. クラスター CLI の使用手順

1. **avro-serialize-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。

2. 以下のコマンドを使用して、アクションを実行します。

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

3.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

3.4. KAMELET ソースファイル

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

第4章 AWS KINESIS SINK

データを AWS Kinesis に送信します。

Kamelet には以下のヘッダーが必要です。

- **partition/ce-partition**: Kinesis のパーティションキーを設定

ヘッダーが設定されていない場合は、エクスチェンジ ID が使用されます。

Kamelet は以下のヘッダーを認識することもできます。

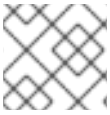
- **sequence-number / ce-sequencenumber**: シーケンス番号を設定します。

このヘッダーは任意です。

4.1. 設定オプション

次の表は、**aws-kinesis-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー	string		
region *	AWS リージョン	以下に接続する AWS リージョン	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー	string		
stream *	ストリーム名	アクセスする Kinesis ストリーム (事前に作成されている必要があります)	string		



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

4.2. DEPENDENCIES

実行時に、**aws-kinesis-sink** Kamelet は以下の依存関係の存在に依存します。

- camel:aws2-kinesis
- camel:kamelet

4.3. 使用方法

ここでは、**aws-kinesis-sink** の使用方法について説明します。

4.3.1. Knative Sink

aws-kinesis-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

4.3.1.2. クラスター CLI の使用手順

1. **aws-kinesis-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

4.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

4.3.2. Kafka Sink

aws-kinesis-sink Kamelet を Kafka シンクとして使用することは、これを Kafka トピックにバインドできます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

4.3.2.2. クラスター CLI の使用手順

1. **aws-kinesis-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

4.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

4.4. KAMELET ソースファイル

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

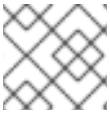
第5章 AWS KINESIS ソース

AWS Kinesis からデータを受信します。

5.1. 設定オプション

次の表は、**aws-kinesis-source** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー	string		
region *	AWS リージョン	以下に接続する AWS リージョン	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー	string		
stream *	ストリーム名	アクセスする Kinesis ストリーム (事前に作成されている必要があります)	string		



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

5.2. DEPENDENCIES

aws-kinesis-source Kamelet は、実行時に以下の依存関係の存在に依存します。

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

5.3. 使用方法

ここでは、**aws-kinesis-source** を使用する方法について説明します。

5.3.1. Knative ソース

aws-kinesis-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

5.3.1.2. クラスター CLI の使用手順

1. **aws-kinesis-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

5.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

5.3.2. Kafka Source

aws-kinesis-source Kamelet を Kafka ソースとして使用するには、これを Kafka トピックにバインドできます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

5.3.2.2. クラスター CLI の使用手順

1. **aws-kinesis-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

5.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

5.4. KAMELET ソースファイル

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

第6章 AWS LAMBDA SINK

AWS Lambda 関数へのペイロードの送信

6.1. 設定オプション

次の表は、**aws-lambda-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー	string		
function *	関数名	Lambda 機能名	string		
region *	AWS リージョン	以下に接続する AWS リージョン	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー	string		

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

6.2. DEPENDENCIES

実行時に、**aws-lambda-sink** Kamelet は以下の依存関係の存在に依存します。

- camel:kamelet
- camel:aws2-lambda

6.3. 使用方法

ここでは、**aws-lambda-sink** の使用方法について説明します。

6.3.1. Knative Sink

aws-lambda-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

6.3.1.2. クラスター CLI の使用手順

1. **aws-lambda-sink-binding.yaml** ファイルをローカルドライブに保存してから、設定に応じて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

6.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

6.3.2. Kafka Sink

aws-lambda-sink Kamelet を Kafka のトピックにバインドすることで、Kafka のシンクとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

6.3.2.2. クラスター CLI の使用手順

1. **aws-lambda-sink-binding.yaml** ファイルをローカルドライブに保存してから、設定に応じて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

6.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

6.4. KAMELET ソースファイル

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

第7章 AWS REDSHIFT シンク

AWS Redshift データベースにデータを送信します。

この Kamelet は JSON をボディーとして想定します。JSON フィールドとパラメーター間のマッピングはキーで実行されるため、以下のクエリーがある場合は以下を実行します。

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

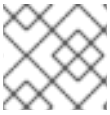
Kamelet は、以下のような入力として受信する必要があります。

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

7.1. 設定オプション

次の表は、**aws-redshift-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
databaseName *	データベース名	ポイントするデータベース名	string		
Password *	Password	セキュリティーで保護された AWS Redshift データベースにアクセスするために使用するパスワード	string		
query *	クエリー	AWS Redshift データベースに対して実行するクエリー	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	サーバー名	データソースのサーバー名	string		**"localhost"**
username *	Username	セキュリティーで保護された AWS Redshift データベースにアクセスするために使用するユーザー名	string		
serverPort	サーバーポート	データソースのサーバーポート	string	**5439**	



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

7.2. DEPENDENCIES

実行時に、**aws-redshift-sink** Kamelet は以下の依存関係の存在に依存します。

- 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

このセクションでは、**aws-redshift-sink** の使用方法について説明します。

7.3.1. Knative Sink

aws-redshift-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

7.3.1.2. クラスター CLI の使用手順

1. **aws-redshift-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

7.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

7.3.2. Kafka Sink

aws-redshift-sink Kamelet を Kafka のトピックにバインドすることで、Kafka のシンクとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

7.3.2.2. クラスター CLI の使用手順

1. **aws-redshift-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

7.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

7.4. KAMELET ソースファイル

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

第8章 AWS SNS SINK

AWS SNS トピックへのメッセージの送信

8.1. 設定オプション

次の表は、**aws-sns-sink** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー	string		
region *	AWS リージョン	以下に接続する AWS リージョン	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー	string		
topicNameOrArn *	トピック名	SQS トピック名または ARN。	string		
autoCreateTopic	トピックの自動作成	SNS トピックの自動作成を設定します。	boolean	**false**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

8.2. DEPENDENCIES

実行時に、**aws-sns-sink** Kamelet は以下の依存関係の存在に依存します。

- camel:kamelet
- camel:aws2-sns

8.3. 使用方法

ここでは、**aws-sns-sink** の使用方法について説明します。

8.3.1. Knative Sink

aws-sns-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

8.3.1.2. クラスター CLI の使用手順

1. **aws-sns-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

8.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

8.3.2. Kafka Sink

aws-sns-sink Kamelet を Kafka シンクとして使用することは、Kafka トピックにバインドできます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

8.3.2.2. クラスター CLI の使用手順

1. **aws-sns-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

8.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

8.4. KAMELET ソースファイル

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

第9章 AWS SQS SINK

メッセージを AWS SQS キューに送信します。

9.1. 設定オプション

次の表は、**aws-sqs-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー	string		
queueNameOrArn *	キュー名	SQS キュー名または ARN。	string		
region *	AWS リージョン	以下に接続する AWS リージョン	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー	string		
autoCreateQueue	自動作成キュー	SQS キューの自動作成の設定。	boolean	**false**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

9.2. DEPENDENCIES

実行時に、**aws-sqs-sink** Kamelet は以下の依存関係の存在に依存します。

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

9.3. 使用方法

ここでは、**aws-sqs-sink** の使用方法について説明します。

9.3.1. Knative Sink

aws-sqs-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

9.3.1.2. クラスター CLI の使用手順

1. **aws-sqs-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

9.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

9.3.2. Kafka Sink

aws-sqs-sink Kamelet を Kafka シンクとして使用することは、これを Kafka トピックにバインドできます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

9.3.2.2. クラスター CLI の使用手順

1. **aws-sqs-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

9.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

9.4. KAMELET ソースファイル

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

第10章 AWS SQS ソース

AWS SQS からデータを受け取ります。

10.1. 設定オプション

次の表は、**aws-sqs-source** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー	string		
queueNameOrArn *	キュー名	SQS キュー名または ARN。	string		
region *	AWS リージョン	以下に接続する AWS リージョン	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー	string		
autoCreateQueue	自動作成キュー	SQS キューの自動作成の設定。	boolean	**false**	
deleteAfterRead	メッセージの自動削除	メッセージの使用後のメッセージの削除	boolean	**true**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

10.2. DEPENDENCIES

実行時には、**aws-sqs-source** Kamelet は以下の依存関係の存在に依存します。

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

10.3. 使用方法

ここでは、**aws-sqs-source** の使用方法について説明します。

10.3.1. Knative ソース

aws-sqs-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

10.3.1.2. クラスター CLI の使用手順

1. **aws-sqs-source-binding.yaml** ファイルをローカルドライブに保存し、設定に応じて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

10.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

10.3.2. Kafka Source

aws-sqs-source Kamelet を Kafka のトピックにバインドすることで、Kafka のソースとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

10.3.2.2. クラスター CLI の使用手順

1. **aws-sqs-source-binding.yaml** ファイルをローカルドライブに保存し、設定に応じて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

10.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

10.4. KAMELET ソースファイル

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 シンク

AWS SQS FIFO キューにメッセージを送信します。

11.1. 設定オプション

次の表は、**aws-sqs-fifo-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー	string		
queueNameOrArn *	キュー名	SQS キュー名または ARN。	string		
region *	AWS リージョン	以下に接続する AWS リージョン	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー	string		
autoCreateQueue	自動作成キュー	SQS キューの自動作成の設定。	boolean	**false**	
contentBasedDeduplication	コンテンツベースの重複排除	コンテンツベースの重複排除を使用 (最初に SQS FIFO キューで有効にする必要があります)	boolean	**false**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

11.2. DEPENDENCIES

実行時に、**aws-sqs-fifo-sink** Kamelet は以下の依存関係の存在に依存します。

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

11.3. 使用方法

ここでは、**aws-sqs-fifo-sink** の使用方法について説明します。

11.3.1. Knative Sink

aws-sqs-fifo-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

11.3.1.2. クラスター CLI の使用手順

1. **aws-sqs-fifo-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

11.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

11.3.2. Kafka Sink

aws-sqs-fifo-sink Kamelet を Kafka シンクとして使用することは、これを Kafka トピックにバインドできます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

11.3.2.2. クラスター CLI の使用手順

1. **aws-sqs-fifo-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

11.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

11.4. KAMELET ソースファイル

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

第12章 AWS S3 SINK

データを AWS S3 にアップロードします。

Kamelet では、以下のヘッダーが設定されていることを想定しています。

- **file/ce-file**: アップロードするファイル名として

ヘッダーが設定されていない場合、エクスチェンジ ID はファイル名として使用されます。

12.1. 設定オプション

次の表は、**aws-s3-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー。	string		
bucketNameOrArn *	バケット名	S3 Bucket 名または ARN。	string		
region *	AWS リージョン	接続する AWS リージョン。	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー。	string		
autoCreateBucket	autocreate バケット	S3 バケット bucketName の自動作成の設定。	boolean	**false**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

12.2. DEPENDENCIES

実行時に、**aws-s3-sink** Kamelet は以下の依存関係の存在に依存します。

- camel:aws2-s3
- camel:kamelet

12.3. 使用方法

ここでは、**aws-s3-sink** の使用方法について説明します。

12.3.1. Knative Sink

aws-s3-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

12.3.1.2. クラスター CLI の使用手順

1. **aws-s3-sink-binding.yaml** ファイルをローカルドライブに保存してから、設定に応じて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

12.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

12.3.2. Kafka Sink

aws-s3-sink Kamelet を Kafka のトピックにバインドすることで、Kafka のシンクとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

12.3.2.2. クラスター CLI の使用手順

1. **aws-s3-sink-binding.yaml** ファイルをローカルドライブに保存してから、設定に応じて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

12.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

12.4. KAMELET ソースファイル

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

第13章 AWS S3 ソース

AWS S3 からデータを受け取ります。

13.1. 設定オプション

次の表は、**aws-s3-source** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー	string		
bucketNameOrArn *	バケット名	S3 バケット名または ARN。	string		
region *	AWS リージョン	以下に接続する AWS リージョン	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー	string		
autoCreateBucket	autocreate バケット	S3 バケット bucketName の自動作成の設定。	boolean	**false**	
deleteAfterRead	自動削除オブジェクト	オブジェクトの使用後のオブジェクトの削除	boolean	**true**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

13.2. DEPENDENCIES

aws-s3-source Kamelet は、実行時に以下の依存関係の存在に依存します。

- camel:kamelet
- camel:aws2-s3

13.3. 使用方法

ここでは、**aws-s3-source** を使用する方法について説明します。

13.3.1. Knative ソース

aws-s3-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

13.3.1.2. クラスター CLI の使用手順

1. **aws-s3-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してソースを実行します。

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

13.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

13.3.2. Kafka Source

aws-s3-source Kamelet を Kafka のトピックにバインドすることで、Kafka のソースとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

13.3.2.2. クラスター CLI の使用手順

1. **aws-s3-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してソースを実行します。

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

13.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

13.4. KAMELET ソースファイル

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

データをストリーミングアップロードモードで AWS S3 にアップロードします。

14.1. 設定オプション

次の表は、**aws-s3-streaming-upload-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
accessKey *	アクセスキー	AWS から取得したアクセスキー。	string		
bucketNameOrArn *	バケット名	S3 Bucket 名または ARN。	string		
keyName *	キー名	endpoint パラメーター経由でバケットの要素のキー名を設定します。Streaming Upload では、デフォルト設定で、ファイルの進捗作成のベースとなります。	string		
region *	AWS リージョン	接続する AWS リージョン。	string		**"eu-west-1"**
secretKey *	シークレットキー	AWS から取得したシークレットキー。	string		
autoCreateBucket	autocreate バケット	S3 バケット bucketName の自動作成の設定。	boolean	**false**	
batchMessageNumber	バッチメッセージ番号	ストリーミングのアップロードモードでバッチを作成するメッセージの数	int	**10**	
batchSize	バッチサイズ	ストリーミングのアップロードモードのバッチサイズ (バイト単位)	int	**1000000**	

プロパティー	名前	説明	型	デフォルト	例
namingStrategy	命名ストラテジー	ストリーミングのアップロードモードで使用する命名ストラテジー。2 つの列挙があり、値は progressive、random のいずれかです。	string	**"progressive"**	
restartingPolicy	ポリシーの再起動	ストリーミングのアップロードモードで使用する再起動ポリシー。2 つの列挙があり、値は override または lastPart の 1 つになります。	string	**"lastPart"**	
streamingUploadMode	ストリーミングアップロードモード	Streaming Upload モードの設定	boolean	**true**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

14.2. DEPENDENCIES

aws-s3-streaming-upload-sink Kamelet は、実行時に以下の依存関係の存在に依存します。

- camel:aws2-s3
- camel:kamelet

14.3. 使用方法

ここでは、**aws-s3-streaming-upload-sink** の使用方法について説明します。

14.3.1. Knative Sink

aws-s3-streaming-upload-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

14.3.1.2. クラスター CLI の使用手順

1. **aws-s3-streaming-upload-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に応じて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

14.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

14.3.2. Kafka Sink

aws-s3-streaming-upload-sink Kamelet を Kafka トピックにバインドすることにより、Kafka シンクとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

14.3.2.2. クラスター CLI の使用手順

1. **aws-s3-streaming-upload-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に応じて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

14.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

14.4. KAMELET ソースファイル

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

第15章 CASSANDRA SINK

データを Cassandra クラスターに送信します。

この Kamelet はボディーを JSON アレイとして想定します。JSON アレイの内容は、クエリーパラメーターに設定された CQL Prepared Statement の入力として使用されます。

15.1. 設定オプション

次の表は、**cassandra-sink** Kamelet で使用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
connectionHost *	接続ホスト	Hostname(s) cassandra server(s). 複数のホストはコンマで区切ることができます。	string		**"localhost"**
connectionPort *	接続ポート	cassandra サーバーのポート番号	string		**9042**
keyspace *	キースペース	使用するキースペース	string		**"customers"**
Password *	Password	セキュアな Cassandra クラスターへのアクセスに使用するパスワード	string		
query *	クエリー	Cassandra クラスターテーブルに対して実行するクエリー	string		
username *	Username	セキュアな Cassandra クラスターへのアクセスに使用するユーザー名	string		
consistencyLevel	一貫性レベル	使用する一貫性レベル。値には、ANY、ONE、TWO、THREE、QUORUM、ALL、LOCAL_QUORUM、EACH_QUORUM、EACH_QUORUM、LOCAL_SERIAL、LOCAL_ONE のいずれかを指定できます。	string	**"ANY"**	

プロパティー	名前	説明	型	デフォルト	例

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

15.2. DEPENDENCIES

実行時に、**cassandra-sink** Kamelet は以下の依存関係の存在に依存します。

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

15.3. 使用方法

ここでは、**cassandra-sink** の使用方法について説明します。

15.3.1. Knative Sink

cassandra-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

15.3.1.2. クラスター CLI の使用手順

1. **cassandra-sink-binding.yaml** ファイルをローカル・ドライブに保存し、設定に合わせて必要な編集を行います。
2. 以下のコマンドを使用してシンクを実行します。

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

15.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

15.3.2. Kafka Sink

cassandra-sink Kamelet を Kafka のトピックにバインドすることで、Kafka のシンクとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

15.3.2.2. クラスター CLI の使用手順

1. **cassandra-sink-binding.yaml** ファイルをローカル・ドライブに保存し、設定に合わせて必要な編集を行います。
2. 以下のコマンドを使用してシンクを実行します。

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

15.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

15.4. KAMELET ソースファイル

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

第16章 CASSANDRA ソース

Cassandra クラスターテーブルをクエリーします。

16.1. 設定オプション

次の表は、**cassandra-source** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
connectionHost *	接続ホスト	Hostname(s) cassandra server(s). 複数のホストはコンマで区切ることができます。	string		**"localhost"**
connectionPort *	接続ポート	cassandra サーバーのポート番号	string		**9042**
keyspace *	キースペース	使用するキースペース	string		**"customers"**
Password *	Password	セキュアな Cassandra クラスターへのアクセスに使用するパスワード	string		
query *	クエリー	Cassandra クラスターテーブルに対して実行するクエリー	string		
username *	Username	セキュアな Cassandra クラスターへのアクセスに使用するユーザー名	string		
consistencyLevel	一貫性レベル	使用する一貫性レベル。値には、ANY、ONE、TWO、THREE、QUORUM、ALL、LOCAL_QUORUM、EACH_QUORUM、EACH_QUORUM、LOCAL_SERIAL、LOCAL_ONE のいずれかを指定できます。	string	**"QUORUM"**	

プロパティー	名前	説明	型	デフォルト	例
resultStrategy	結果ストラテジー	クエリーの結果セットを変換するストラテジー。使用できる値は ALL、ONE、LIMIT_10、LIMIT_100...	string	**"ALL"**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

16.2. DEPENDENCIES

起動時に、**cassandra-source** Kamelet は、以下の依存関係の存在に依存します。

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

16.3. 使用方法

本セクションでは、**cassandra-source** を使用する方法を説明します。

16.3.1. Knative ソース

cassandra-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

16.3.1.2. クラスター CLI の使用手順

1. **cassandra-source-binding.yaml** ファイルをローカル・ドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用してソースを実行します。

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

16.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

16.3.2. Kafka Source

cassandra-source Kamelet を Kafka トピックにバインドすることで、Kafka ソースとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

16.3.2.2. クラスター CLI の使用手順

1. **cassandra-source-binding.yaml** ファイルをローカル・ドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用してソースを実行します。

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

16.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

16.4. KAMELET ソースファイル

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

第17章 フィールドアクションの抽出

本文からフィールドを抽出します

17.1. 設定オプション

以下の表では、**extract-field-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
field *	フィールド	追加するフィールドの名前	string		

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

17.2. DEPENDENCIES

実行時に、**extract-field-action** Kamelet は以下の依存関係の存在に依存します。

- 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

このセクションでは、**extract-field-action** の使用方法について説明します。

17.3.1. Knative Action

extract-field-action Kamelet は、Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

17.3.1.2. クラスター CLI の使用手順

1. **extract-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

17.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

17.3.2. Kafka Action

extract-field-action Kamelet は、Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

17.3.2.2. クラスター CLI の使用手順

1. **extract-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

17.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

17.4. KAMELET ソースファイル

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

第18章 FTP SINK

データを FTP サーバーに送信します。

Kamelet では、以下のヘッダーが設定されていることを想定しています。

- **file/ce-file**: アップロードするファイル名として

ヘッダーが設定されていない場合、エクスチェンジ ID はファイル名として使用されます。

18.1. 設定オプション

次の表は、**ftp-sink** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
connectionHost *	接続ホスト	FTP サーバーのホスト名	string		
connectionPort *	接続ポート	FTP サーバーのポート	string	**21**	
directoryName *	ディレクトリー名	開始ディレクトリー	string		
Password *	Password	FTP サーバーにアクセスするためのパスワード	string		
username *	Username	FTP サーバーにアクセスするためのユーザー名	string		
fileExist	ファイルの存在	すでにファイルが存在する場合にどのように動作するか。列挙は 4 つあり、値は Override、Append、Fail または Ignore のいずれかです。	string	**"Override"**	
passiveMode	パッシブモード	パッシブモード接続の設定	boolean	**false**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

18.2. DEPENDENCIES

起動時に、**ftp-sink** Kamelet は、以下の依存関係の存在に依存します。

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

18.3. 使用方法

本セクションでは、**ftp-sink** を使用する方法を説明します。

18.3.1. Knative Sink

ftp-sink Kamelet を Knative オブジェクトにバインドすることにより、Knative シンクとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

18.3.1.2. クラスター CLI の使用手順

1. **ftp-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

18.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

18.3.2. Kafka Sink

ftp-sink Kamelet を Kafka トピックにバインドすることにより、Kafka シンクとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

18.3.2.2. クラスター CLI の使用手順

1. **ftp-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

18.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

18.4. KAMELET ソースファイル

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

第19章 FTP ソース

FTP サーバーからデータを受信します。

19.1. 設定オプション

以下の表は、**ftp-source** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
connectionHost *	接続ホスト	FTP サーバーのホスト名	string		
connectionPort *	接続ポート	FTP サーバーのポート	string	**21**	
directoryName *	ディレクトリー名	開始ディレクトリー	string		
Password *	Password	FTP サーバーにアクセスするためのパスワード	string		
username *	Username	FTP サーバーにアクセスするためのユーザー名	string		
idempotent	冪等性	処理されたファイルを省略します。	boolean	**true**	
passiveMode	パッシブモード	パッシブモード接続の設定	boolean	**false**	
再帰	再帰	ディレクトリーの場合は、すべてのサブディレクトリー内のファイルも検索します。	boolean	**false**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

19.2. DEPENDENCIES

起動時に、**ftp-source** Kamelet は、以下の依存関係の存在に依存します。

- camel:ftp

- camel:core
- camel:kamelet

19.3. 使用方法

本セクションでは、**ftp-source** を使用する方法を説明します。

19.3.1. Knative ソース

ftp-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

19.3.1.2. クラスター CLI の使用手順

1. **ftp-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してソースを実行します。

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

19.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

19.3.2. Kafka Source

ftp-source Kamelet を Kafka のトピックにバインドすることで、Kafka のソースとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

19.3.2.2. クラスター CLI の使用手順

1. **ftp-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してソースを実行します。

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

19.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

19.4. KAMELET ソースファイル

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

第20章 HAS HEADER FILTER ACTION

1 つのヘッダーの存在に基づくフィルター

20.1. 設定オプション

以下の表では、**has-header-filter-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
name *	ヘッダー名	評価するヘッダー名。ヘッダー名は、ソース Kamelet から渡す必要があります。Knative の場合のみ、クラウドイベントを使用している場合は、ヘッダー名に CloudEvent (ce-) 接頭辞を含める必要があります。	string		**"headerName"**

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

20.2. DEPENDENCIES

ランタイム時に、**has-header-filter-action** Kamelet は、以下の依存関係の存在に依存します。

- camel:core
- camel:kamelet

20.3. 使用方法

本セクションでは、**has-header-filter-action** の使用方法を説明します。

20.3.1. Knative Action

has-header-filter-action Kamelet を Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

20.3.1.2. クラスター CLI の使用手順

1. **has-header-filter-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

20.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

20.3.2. Kafka Action

has-header-filter-action Kamelet を Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

20.3.2.2. クラスター CLI の使用手順

1. **has-header-filter-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

20.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

20.4. KAMELET ソースファイル

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

第21章 HOIST フィールドアクション

データを単一のフィールドにラップします。

21.1. 設定オプション

以下の表では、**hoist-field-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
field *	フィールド	イベントを含むフィールドの名前	string		

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

21.2. DEPENDENCIES

実行時に、**hoist-field-action** Kamelet は以下の依存関係の存在に依存します。

- 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

このセクションでは、**hoist-field-action** の使用方法について説明します。

21.3.1. Knative Action

hoist-field-action Kamelet は、Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

21.3.1.2. クラスター CLI の使用手順

1. **hoist-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

21.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

21.3.2. Kafka Action

hoist-field-action Kamelet は、Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

21.3.2.2. クラスター CLI の使用手順

1. **hoist-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

21.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

21.4. KAMELET ソースファイル

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

第22章 HTTP SINK

イベントの HTTP エンドポイントへの転送

22.1. 設定オプション

次の表は、**http-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
url *	URL	データの送信先となる URL	string		**"https://my-service/path"**
メソッド	メソッド	使用する HTTP メソッド	string	**"POST"**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

22.2. DEPENDENCIES

実行時には、**http-sink** Kamelet は以下の依存関係の存在に依存します。

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

22.3. 使用方法

ここでは、**http-sink** の使用方法について説明します。

22.3.1. Knative Sink

http-sink Kamelet を Knative オブジェクトにバインドし、Knative シンクとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

22.3.1.2. クラスター CLI の使用手順

1. **http-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

22.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

22.3.2. Kafka Sink

http-sink Kamelet を Kafka トピックにバインドすることにより、Kafka シンクとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

22.3.2.2. クラスター CLI の使用手順

1. **http-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

22.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

22.4. KAMELET ソースファイル

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

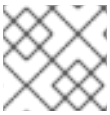
第23章 フィールドアクションの挿入

転送中のメッセージに定数値を持つカスタムフィールドを追加します。

23.1. 設定オプション

以下の表では、**insert-field-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
field *	フィールド	追加するフィールドの名前	string		
value *	値	フィールドの値	string		



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

23.2. DEPENDENCIES

実行時に、**insert-field-action** Kamelet は以下の依存関係の存在に依存します。

- 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

ここでは、**insert-field-action** の使用方法について説明します。

23.3.1. Knative Action

insert-field-action Kamelet は、Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

23.3.1.2. クラスター CLI の使用手順

1. **insert-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

23.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

23.3.2. Kafka Action

insert-field-action Kamelet は、Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

23.3.2.2. クラスター CLI の使用手順

1. **insert-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

23.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

23.4. KAMELET ソースファイル

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

第24章 ヘッダーアクションの挿入

転送中のメッセージに定数値を持つヘッダーを追加します。

24.1. 設定オプション

以下の表では、**insert-header-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
name *	名前	追加するヘッダーの名前Knative の場合のみ、ヘッダーの名前には CloudEvent (ce-) 接頭辞が必要です。	string		
value *	値	ヘッダーの値	string		

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

24.2. DEPENDENCIES

ランタイム時に、**insert-header-action** Kamelet は、以下の依存関係の存在に依存します。

- camel:core
- camel:kamelet

24.3. 使用方法

本セクションでは、**insert-header-action** の使用方法を説明します。

24.3.1. Knative Action

insert-header-action Kamelet を Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

24.3.1.2. クラスター CLI の使用手順

1. **insert-header-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

24.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

24.3.2. Kafka Action

insert-header-action Kamelet を Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

24.3.2.2. クラスター CLI の使用手順

1. **insert-header-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

24.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

24.4. KAMELET ソースファイル

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

第25章 IS TOMBSTONE FILTER ACTION

ボディーの存在に基づくフィルター

25.1. 設定オプション

is-tombstone-filter-action Kamelet は設定オプションを指定しません。

25.2. DEPENDENCIES

実行時、**is-tombstone-filter-action** Kamelet は、以下の依存関係の存在に依存しています。

- camel:core
- camel:kamelet

25.3. 使用方法

ここでは、**is-tombstone-filter-action** の使用方法について説明します。

25.3.1. Knative Action

is-tombstone-filter-action Kamelet は、Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

25.3.1.2. クラスター CLI の使用手順

1. **is-tombstone-filter-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

25.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

25.3.2. Kafka Action

is-tombstone-filter-action Kamelet を Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

25.3.2.2. クラスター CLI の使用手順

1. **is-tombstone-filter-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

25.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

25.4. KAMELET ソースファイル

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

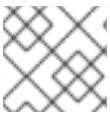
第26章 JIRA ソース

Jira から新しい問題に関する通知を受け取ります。

26.1. 設定オプション

次の表は、**jira-source** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
jiraUrl *	Jira URL	Jira インスタンスの URL	string		**"http://my_jira.com:8081"**
Password *	Password	Jira にアクセスするためのパスワード	string		
username *	Username	Jira にアクセスするためのユーザー名	string		
jql	JQL	問題をフィルターするクエリー	string		**"project=MyProject"**



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

26.2. DEPENDENCIES

ランタイム時に、**jira-source** Kamelet は以下の依存関係の存在に依存します。

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

26.3. 使用方法

ここでは、**jira-source** の使用方法について説明します。

26.3.1. Knative ソース

jira-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

26.3.1.2. クラスター CLI の使用手順

1. **jira-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用してソースを実行します。

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

26.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

26.3.2. Kafka Source

jira-source Kamelet を Kafka トピックにバインドすることにより、Kafka のソースとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

26.3.2.2. クラスター CLI の使用手順

1. **jira-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用してソースを実行します。

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

26.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

26.4. KAMELET ソースファイル

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

Apache Qpid JMS クライアントを使用して、任意の AMQP 1.0 準拠のメッセージブローカーにイベントを生成できる Kamelet

27.1. 設定オプション

次の表は、**jms-amqp-10-sink** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
destinationName *	宛先名	JMS 宛先名	string		
remoteURI *	ブローカーの URL	JMS URL	string		**"amqp://my-host:31616"**
destinationType	宛先タイプ	JMS 宛先タイプ (例: キューまたはトピック)	string	**"queue"**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

27.2. DEPENDENCIES

jms-amqp-10-sink Kamelet は、実行時に以下の依存関係の存在に依存します。

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

27.3. 使用方法

本セクションでは、jms **-amqp-10-sink** の使用方法を説明します。

27.3.1. Knative Sink

jms-amqp-10-sink Kamelet を Knative オブジェクトにバインドし、これを Knative シンクとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

27.3.1.2. クラスター CLI の使用手順

1. **jms-amqp-10-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してシンクを実行します。

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

27.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

27.3.2. Kafka Sink

jms-amqp-10-sink Kamelet を Kafka シンクとして使用することは、Kafka トピックにバインドします。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

27.3.2.2. クラスター CLI の使用手順

1. **jms-amqp-10-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してシンクを実行します。

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

27.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

27.4. KAMELET ソースファイル

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 ソース

Apache Qpid JMS クライアントを使用して、どの AMQP 1.0 準拠メッセージブローカーからのイベントを消費できる Kamelet

28.1. 設定オプション

次の表は、**jms-amqp-10-source** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
destinationName *	宛先名	JMS 宛先名	string		
remoteURI *	ブローカーの URL	JMS URL	string		**"amqp://my-host:31616"**
destinationType	宛先タイプ	JMS 宛先タイプ (例: キューまたはトピック)	string	**"queue"**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

28.2. DEPENDENCIES

jms-amqp-10-source Kamelet は、実行時に以下の依存関係の存在に依存します。

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

28.3. 使用方法

ここでは、**jms-amqp-10-source** の使用方法について説明します。

28.3.1. Knative ソース

jms-amqp-10-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

28.3.1.2. クラスター CLI の使用手順

1. **jms-amqp-10-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してソースを実行します。

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

28.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

28.3.2. Kafka Source

jms-amqp-10-source Kamelet を Kafka トピックにバインドすることにより、Kafka ソースとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

28.3.2.2. クラスター CLI の使用手順

1. **jms-amqp-10-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してソースを実行します。

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

28.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

28.4. KAMELET ソースファイル

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 シンク

JMS を使用して IBM MQ メッセージキューにイベントを生成できる Kamelet。

29.1. 設定オプション

次の表は、**jms-ibm-mq-sink** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
channel *	IBM MQ Channel	IBM MQ チャネルの名前	string		
destinationName *	宛先名	宛先名	string		
Password *	Password	IBM MQ サーバーに対して認証するためのパスワード	string		
queueManager *	IBM MQ Queue Manager	IBM MQ Queue Manager の名前	string		
serverName *	IBM MQ Server サーバー名	IBM MQ Server の名前またはアドレス	string		
serverPort *	IBM MQ Server ポート	IBM MQ Server ポート	integer	**1414**	
username *	Username	IBM MQ サーバーに対して認証するためのユーザー名	string		
clientId	IBM MQ Client ID	IBM MQ クライアント ID の名前	string		
destinationType	宛先タイプ	JMS 宛先タイプ (キューまたはトピック)	string	**"queue"**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

29.2. DEPENDENCIES

jms-ibm-mq-sink Kamelet は、実行時に以下の依存関係の存在に依存します。

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

29.3. 使用方法

このセクションでは、**jms-ibm-mq-sink** の使用方法について説明します。

29.3.1. Knative Sink

jms-ibm-mq-sink Kamelet を Knative オブジェクトにバインドし、これを Knative シンクとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

29.3.1.2. クラスター CLI の使用手順

1. **jms-ibm-mq-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。

2. 以下のコマンドを使用してシンクを実行します。

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

29.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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'
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

29.3.2. Kafka Sink

jms-ibm-mq-sink Kamelet を Kafka シンクとして使用することは、Kafka トピックにバインドします。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

29.3.2.2. クラスター CLI の使用手順

1. **jms-ibm-mq-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。

2. 以下のコマンドを使用してシンクを実行します。

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

29.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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'
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

29.4. KAMELET ソースファイル

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 ソース

JMS を使用して IBM MQ メッセージキューからイベントを読み取ることができる Kamelet。

30.1. 設定オプション

次の表は、**jms-ibm-mq-source** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
channel *	IBM MQ Channel	IBM MQ チャネルの名前	string		
destinationName *	宛先名	宛先名	string		
Password *	Password	IBM MQ サーバーに対して認証するためのパスワード	string		
queueManager *	IBM MQ Queue Manager	IBM MQ Queue Manager の名前	string		
serverName *	IBM MQ Server サーバー名	IBM MQ Server の名前またはアドレス	string		
serverPort *	IBM MQ Server ポート	IBM MQ Server ポート	integer	**1414**	
username *	Username	IBM MQ サーバーに対して認証するためのユーザー名	string		
clientId	IBM MQ Client ID	IBM MQ クライアント ID の名前	string		
destinationType	宛先タイプ	JMS 宛先タイプ (キューまたはトピック)	string	**"queue"**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

30.2. DEPENDENCIES

jms-ibm-mq-source Kamelet は、実行時に以下の依存関係の存在に依存します。

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

30.3. 使用方法

このセクションでは、**jms-ibm-mq-source** の使用方法について説明します。

30.3.1. Knative ソース

jms-ibm-mq-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

30.3.1.2. クラスター CLI の使用手順

1. **jms-ibm-mq-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。

2. 以下のコマンドを使用してソースを実行します。

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

30.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

30.3.2. Kafka Source

jms-ibm-mq-source Kamelet を Kafka トピックにバインドすることにより、Kafka ソースとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

30.3.2.2. クラスター CLI の使用手順

1. **jms-ibm-mq-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してソースを実行します。

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

30.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

30.4. KAMELET ソースファイル

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

第31章 JSON デシリアライズアクション

ペイロードを JSON にデシリアライズ

31.1. 設定オプション

json-deserialize-action Kamelet は設定オプションを指定しません。

31.2. DEPENDENCIES

json-deserialize-action Kamelet は、実行時に以下の依存関係の存在に依存しています。

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

31.3. 使用方法

ここでは、**Json-deserialize-action** の使い方を説明します。

31.3.1. Knative Action

json-deserialize-action Kamelet は、Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

31.3.1.2. クラスター CLI の使用手順

1. **json-deserialize-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用して、アクションを実行します。

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

31.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

31.3.2. Kafka Action

json-deserialize-action Kamelet は、Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

31.3.2.2. クラスター CLI の使用手順

1. **json-deserialize-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用して、アクションを実行します。

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

31.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

31.4. KAMELET ソースファイル

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

第32章 JSON シリアルアクション

ペイロードを JSON にシリアライズする

32.1. 設定オプション

json-serialize-action Kamelet は設定オプションを指定しません。

32.2. DEPENDENCIES

実行時に、**json-serialize-action** Kamelet は以下の依存関係の存在に依存します。

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

32.3. 使用方法

ここでは、**json-serialize-action** の使い方を説明します。

32.3.1. Knative Action

json-serialize-action Kamelet は、Knative のバインディングの中間ステップとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

32.3.1.2. クラスター CLI の使用手順

1. **json-serialize-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用して、アクションを実行します。

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

32.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

32.3.2. Kafka Action

json-serialize-action Kamelet は、Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

32.3.2.2. クラスター CLI の使用手順

1. **json-serialize-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用して、アクションを実行します。

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

32.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

32.4. KAMELET ソースファイル

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

第33章 KAFKA SINK

データを Kafka トピックに送信します。

Kamelet は、設定するヘッダーについて理解することができます。

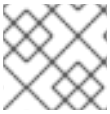
- **key/ce-key**: メッセージキーとして
- **partition-key/ce-partitionkey**: メッセージパーティションキーとして

ヘッダーはいずれもオプションです。

33.1. 設定オプション

次の表は、**kafka-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
bootstrapServers *	ブローカー	Kafka Broker URL のコンマ区切りリスト	string		
Password *	Password	kafka に対して認証を行うためのパスワード	string		
Topic *	トピック名	Kafka トピック名のコンマ区切りリスト	string		
user *	Username	Kafka に対して認証を行うためのユーザー名	string		
saslMechanism	SASL メカニズム	使用される Simple Authentication and Security Layer(SASL) メカニズム。	string	**"PLAIN"**	
securityProtocol	セキュリティープロトコル	ブローカーとの通信に使用されるプロトコル。SASL_PLAINTEXT、PLAINTEXT、SASL_SSL、および SSL がサポートされます。	string	**"SASL_SSL"**	



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

33.2. DEPENDENCIES

kafka-sink Kamelet は、以下の依存関係の存在に依存します。

- camel:kafka
- camel:kamelet

33.3. 使用方法

ここでは、**kafka-sink** の使用方法について説明します。

33.3.1. Knative Sink

kafka-sink Kamelet を Knative のオブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

33.3.1.2. クラスター CLI の使用手順

1. **kafka-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してシンクを実行します。

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

33.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

33.3.2. Kafka Sink

kafka-sink Kamelet を Kafka トピックにバインドすることにより、Kafka シンクとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

33.3.2.2. クラスター CLI の使用手順

1. **kafka-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。

2. 以下のコマンドを使用してシンクを実行します。

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

33.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

33.4. KAMELET ソースファイル

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

第34章 KAFKA SOURCE

Kafka トピックからデータを受信します。

34.1. 設定オプション

次の表は、**kafka-source** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
Topic *	トピック名	Kafka トピック名のコンマ区切りリスト	string		
bootstrapServers *	ブローカー	Kafka Broker URL のコンマ区切りリスト	string		
securityProtocol	セキュリティープロトコル	ブローカーとの通信に使用されるプロトコル。SASL_PLAINTEXT、PLAINTEXT、SASL_SSL、および SSL がサポートされます。	string	**"SASL_SSL"**	
saslMechanism	SASL メカニズム	使用される Simple Authentication and Security Layer(SASL) メカニズム。	string	**"PLAIN"**	
user *	Username	Kafka に対して認証を行うためのユーザー名	string		
Password *	Password	kafka に対して認証を行うためのパスワード	string		
autoCommitEnable	自動コミットの有効化	true の場合、コンシューマーによってすでにフェッチされているメッセージのオフセットを ZooKeeper に定期的にコミットします。	boolean	**true**	
allowManualCommit	手動コミットを許可する	手動コミットを許可するかどうか。	boolean	**false**	

プロパティー	名前	説明	型	デフォルト	例
autoOffsetReset	自動オフセットリセット	初期オフセットがない場合のアクション。列挙は 3 つあり、値は latest、earliest、none のいずれかです。	string	**"latest"**	
pollOnError	poll On エラー動作	新しいメッセージのポーリング中に、kafka が例外を出力した場合のアクション。5 つの列挙があり、値は DISCARD、ERROR_HANDLER、RECONNECT、RETRY、STOP のいずれかです。	string	**"ERROR_HANDLER"**	
deserializeHeaders	ヘッダーを自動的に逆シリアル化	有効にすると、Kamelet ソースはすべてのメッセージヘッダーを文字列表現に逆シリアル化します。デフォルトは **false** です。	boolean	**true**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

34.2. DEPENDENCIES

実行時に、`kafka-source Kamelet は以下の依存関係の存在に依存します。

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

34.3. 使用方法

ここでは、**kafka-source** の使用方法について説明します。

34.3.1. Knative ソース

kafka-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

34.3.1.2. クラスター CLI の使用手順

1. **kafka-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

34.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

34.3.2. Kafka Source

kafka-source Kamelet を Kafka トピックにバインドすることにより、Kafka のソースとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

34.3.2.2. クラスター CLI の使用手順

1. **kafka-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

34.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

34.4. KAMELET ソースファイル

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

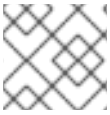
第35章 KAFKA TOPIC NAME の FILTER アクション

正規表現と比較した kafka トピック値に基づくフィルター

35.1. 設定オプション

次の表は、**topic-nam-matches-filter-action** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
regex *	Regex	Kafka トピック名に対して評価する Regex	string		



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

35.2. DEPENDENCIES

ランタイム時に、**topic-name-matches-filter-action** Kamelet は、以下の依存関係の存在に依存します。

- camel:core
- camel:kamelet

35.3. 使用方法

本セクションでは、**topic-name-matches-filter-action** を使用する方法を説明します。

35.3.1. Kafka Action

topic-name-matches-filter-action Kamelet を Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

35.3.1.2. クラスター CLI の使用手順

1. **topic-name-matches-filter-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

35.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

35.4. KAMELET ソースファイル

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

第36章 LOG SINK

受信するすべてのデータをログに記録するシンクは、デバッグに役立ちます。

36.1. 設定オプション

以下の表は、**log-sink** Kamelet で利用可能な設定オプションの概要を示しています。

プロパティー	名前	説明	型	デフォルト	例
showHeaders	show Headers	受信したヘッダーを表示します。	boolean	**false**	
showStreams	Streams を表示	ストリーム本文を表示します (以下の手順では利用できない場合があります)。	boolean	**false**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

36.2. DEPENDENCIES

ランタイム時に、**log-sink** Kamelet は、以下の依存関係の存在に依存します。

- camel:kamelet
- camel:log

36.3. 使用方法

本セクションでは、**log-sink** を使用する方法について説明します。

36.3.1. Knative Sink

log-sink Kamelet を Knative オブジェクトにバインドし、Knative シンクとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

36.3.1.2. クラスター CLI の使用手順

1. **log-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

36.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

36.3.2. Kafka Sink

log-sink Kamelet を Kafka トピックにバインドすることにより、Kafka シンクとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

36.3.2.2. クラスター CLI の使用手順

1. **log-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

36.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

36.4. KAMELET ソースファイル

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

第37章 MARIADB シンク

MariaDB データベースにデータを送信します。

この Kamelet は JSON をボディーとして想定します。JSON フィールドとパラメーター間のマッピングはキーで実行されるため、以下のクエリーがある場合は以下を実行します。

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

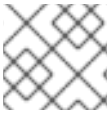
Kamelet は、以下のような入力として受信する必要があります。

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

37.1. 設定オプション

以下の表では、**mariadb-sink** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
databaseName *	データベース名	ポイントするデータベース名	string		
Password *	Password	セキュリティーで保護された MariaDB データベースにアクセスするために使用するパスワード	string		
query *	クエリー	MariaDB データベースに対して実行するクエリー	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	サーバー名	データソースのサーバー名	string		**"localhost"**
username *	Username	セキュリティーで保護された MariaDB データベースにアクセスするために使用するユーザー名	string		
serverPort	サーバーポート	データソースのサーバーポート	string	**3306**	



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

37.2. DEPENDENCIES

ランタイム時に、**mariadb-sink** Kamelet は、以下の依存関係の存在に依存します。

- 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

このセクションでは、**mariadb-sink** の使用方法について説明します。

37.3.1. Knative Sink

mariadb-sink Kamelet を Knative オブジェクトにバインドし、これを Knative シンクとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

37.3.1.2. クラスター CLI の使用手順

1. **mariadb-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。

2. 以下のコマンドを使用してシンクを実行します。

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

37.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

37.3.2. Kafka Sink

mariadb-sink Kamelet を Kafka トピックにバインドすることにより、Kafka シンクとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

37.3.2.2. クラスター CLI の使用手順

1. **mariadb-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

37.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

37.4. KAMELET ソースファイル

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

第38章 MASK フィールドアクション

送信中のメッセージで定数値を持つフィールドをマスクします

38.1. 設定オプション

以下の表では、**mask-field-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
fields *	フィールド	マスクするフィールドのコンマ区切りリスト	string		
replacement *	replacement	マスクするフィールドの置き換え	string		

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

38.2. DEPENDENCIES

実行時に、**mask-field-action** Kamelet は以下の依存関係の存在に依存します。

- 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

このセクションでは、**mask-field-action** の使用方法について説明します。

38.3.1. Knative Action

mask-field-action Kamelet は、Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

38.3.1.2. クラスター CLI の使用手順

1. **mask-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

38.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

38.3.2. Kafka Action

mask-field-action Kamelet は、Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

38.3.2.2. クラスター CLI の使用手順

1. **mask-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

38.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

38.4. KAMELET ソースファイル

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

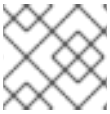
第39章 MESSAGE TIMESTAMP ルーターアクション

topic フィールドを、元のトピック名およびレコードのタイムスタンプフィールドとして更新します。

39.1. 設定オプション

次の表は、**message-timestamp-router-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
timestampKeys *	タイムスタンプキー	Timestamp キーのコンマ区切りリスト。タイムスタンプは最初に見つかったフィールドから取得されます。	string		
timestampFormat	タイムスタンプの形式	java.text.SimpleDateFormat と互換性があるタイムスタンプの文字列の文字列。	string	**"yyyyMMdd"**	
timestampKeyFormat	タイムスタンプキーの形式	タイムスタンプキーの形式。使用できる値は 'timestamp' または java.text.SimpleDateFormat と互換性のあるタイムスタンプの文字列です。'timestamp' の場合、フィールドは 1970 からミリ秒として評価され、UNIX Timestamp として評価されます。	string	**"timestamp"**	
topicFormat	トピックの形式	それぞれトピックとタイムスタンプのプレースホルダーとして '$[topic]' および '$[timestamp]' を含む文字列のフォーマット文字列。	string	**"topic-$[timestamp]"**	



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

39.2. DEPENDENCIES

実行時、**message-timestamp-router-action** Kamelet は、以下の依存関係の存在に依存しています。

- 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

本セクションでは、**message-timestamp-router-action** を使用する方法を説明します。

39.3.1. Knative Action

message-timestamp-router-action Kamelet を Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

39.3.1.2. クラスター CLI の使用手順

1. **message-timestamp-router-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

39.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

39.3.2. Kafka Action

message-timestamp-router-action Kamelet を Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

39.3.2.2. クラスター CLI の使用手順

1. **message-timestamp-router-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

39.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

39.4. KAMELET ソースファイル

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

第40章 MONGODB SINK

MongoDB にドキュメントを送信します。

この Kamelet は JSON をボディーとして想定します。

ヘッダーとして設定できるプロパティー:

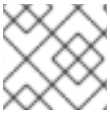
- **db-upsert** / **ce-dbupsert**: データベースが存在しない場合には、この要素を作成します。ブール値。

40.1. 設定オプション

以下の表では、**mongodb-sink** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
collection *	MongoDB コレクション	このエンドポイントにバインドする MongoDB コレクションの名前を設定します。	string		
Database *	MongoDB Database	ターゲットに設定する MongoDB データベースの名前を設定します。	string		
hosts *	MongoDB Hosts	host:port 形式の MongoDB ホストアドレスのコンマ区切りリスト。	string		
createCollection	コレクション	コレクションが存在しない場合は、初期化中にコレクションを作成します。	boolean	**false**	
password	MongoDB パスワード	MongoDB にアクセスするためのユーザーパスワード。	string		
username	MongoDB ユーザー名	MongoDB にアクセスするためのユーザー名	string		

プロパティー	名前	説明	型	デフォルト	例
writeConcern	書き込みに関する懸念	書き込み操作用に MongoDB から要求される確認応答のレベルを設定します。可能な値は、ACKNOWLEDGED、W1、W2、W3、W3、UNACKNOWLEDGED、JOURNALED、MAJORITY です。	string		



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

40.2. DEPENDENCIES

ランタイム時に、**mongodb-sink** Kamelet は、以下の依存関係の存在に依存します。

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

40.3. 使用方法

本セクションでは、**mongodb-sink** を使用する方法を説明します。

40.3.1. Knative Sink

mongodb-sink Kamelet を Knative オブジェクトにバインドし、これを Knative シンクとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

40.3.1.2. クラスター CLI の使用手順

1. **mongodb-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

40.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

40.3.2. Kafka Sink

mongodb-sink Kamelet を Kafka トピックにバインドすることにより、Kafka シンクとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

40.3.2.2. クラスター CLI の使用手順

1. **mongodb-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

40.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

40.4. KAMELET ソースファイル

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

第41章 MONGODB ソース

MongoDB からドキュメントを消費します。

persistentTailTracking オプションを有効にすると、コンシューマーは最後に消費されるメッセージを追跡し、次の再起動時に、そのメッセージから消費が再起動します。persistentTailTracking が有効にされている場合、tailTrackIncreasingField を指定する必要があります (デフォルトではオプションです)。

persistentTailTracking オプションが有効になっていない場合、コンシューマーはコレクション全体を消費し、新しいドキュメントが消費するアイドル状態になるのを待ちます。

41.1. 設定オプション

以下の表では、**mongodb-source** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
collection *	MongoDB コレクション	このエンドポイントにバインドする MongoDB コレクションの名前を設定します。	string		
Database *	MongoDB Database	ターゲットに設定する MongoDB データベースの名前を設定します。	string		
hosts *	MongoDB Hosts	host:port 形式の MongoDB ホストアドレスのコンマ区切りリスト。	string		
Password *	MongoDB パスワード	MongoDB にアクセスするためのユーザーパスワード。	string		
username *	MongoDB ユーザー名	MongoDB にアクセスするためのユーザー名ユーザー名は MongoDB の認証データベース (authenticationDatabase) に存在する必要があります。デフォルトでは、MongoDB authenticationDatabase は admin です。	string		

プロパティー	名前	説明	型	デフォルト	例
persistentTailTracking	MongoDB Persistent Tail Tracking	永続的な tail トラッキングを有効にします。これは、システムの再起動時に最後に消費されたメッセージを追跡するメカニズムです。次にシステムが起動すると、エンドポイントは最後にレコードを一気に読み込むのを停止した地点からカーソルを回復します。	boolean	**false**	
tailTrackIncreasingField	MongoDB Tail Track Increasing フィールド	増加する性質の着信レコードの相関フィールドであり、生成されるたびに tail カーソルを配置するために使用されます。	string		

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

41.2. DEPENDENCIES

ランタイム時に、**mongodb-source** Kamelet は、以下の依存関係の存在に依存します。

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

41.3. 使用方法

本セクションでは、**mongodb-source** の使用方法を説明します。

41.3.1. Knative ソース

mongodb-source Kamelet を Knative オブジェクトにバインドして Knative ソースとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

41.3.1.2. クラスター CLI の使用手順

1. **mongodb-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してソースを実行します。

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

41.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

41.3.2. Kafka Source

mongodb-source Kamelet を Kafka トピックにバインドすることにより、Kafka ソースとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

41.3.2.2. クラスター CLI の使用手順

1. **mongodb-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してソースを実行します。

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

41.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

41.4. KAMELET ソースファイル

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

第42章 MYSQL SINK

MySQL データベースにデータを送信します。

この Kamelet は JSON をボディーとして想定します。JSON フィールドとパラメーター間のマッピングはキーで実行されるため、以下のクエリーがある場合は以下を実行します。

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

Kamelet は、以下のような入力として受信する必要があります。

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

42.1. 設定オプション

次の表は、**mysql-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
databaseName *	データベース名	ポイントするデータベース名	string		
Password *	Password	セキュアな MySQL データベースへのアクセスに使用するパスワード	string		
query *	クエリー	MySQL データベースに対して実行するクエリー	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	サーバー名	データソースのサーバー名	string		**"localhost"**
username *	Username	セキュアな MySQL データベースへのアクセスに使用するユーザー名	string		
serverPort	サーバーポート	データソースのサーバーポート	string	**3306**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

42.2. DEPENDENCIES

実行時に、**mysql-sink** Kamelet は以下の依存関係の存在に依存します。

- 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

ここでは、**mysql-sink** の使用方法について説明します。

42.3.1. Knative Sink

mysql-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

42.3.1.2. クラスター CLI の使用手順

1. **mysql-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。

2. 以下のコマンドを使用してシンクを実行します。

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

42.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

42.3.2. Kafka Sink

mysql-sink Kamelet を Kafka のトピックにバインドすることで、Kafka のシンクとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

42.3.2.2. クラスター CLI の使用手順

1. **mysql-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。

2. 以下のコマンドを使用してシンクを実行します。

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

42.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

42.4. KAMELET ソースファイル

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

第43章 POSTGRESQL SINK

データを PostgreSQL データベースに送信します。

この Kamelet は JSON をボディーとして想定します。JSON フィールドとパラメーター間のマッピングはキーで実行されるため、以下のクエリーがある場合は以下を実行します。

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

Kamelet は、以下のような入力として受信する必要があります。

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

43.1. 設定オプション

次の表は、**postgresql-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
databaseName *	データベース名	ポイントするデータベース名	string		
Password *	Password	セキュアな PostgreSQL データベースへのアクセスに使用するパスワード	string		
query *	クエリー	PostgreSQL データベースに対して実行するクエリー	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	サーバー名	データソースのサーバー名	string		**"localhost"**
username *	Username	セキュアな PostgreSQL データベースへのアクセスに使用するユーザー名	string		
serverPort	サーバーポート	データソースのサーバーポート	string	**5432**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

43.2. DEPENDENCIES

実行時に、**postgresql-sink** Kamelet は以下の依存関係の存在に依存しています。

- 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

ここでは、**postgresql-sink** の使用方法について説明します。

43.3.1. Knative Sink

postgresql-sink Kamelet を Knative オブジェクトにバインドすることで、Knative のシンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

43.3.1.2. クラスター CLI の使用手順

1. **postgresql-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してシンクを実行します。

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

43.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

43.3.2. Kafka Sink

postgresql-sink Kamelet を Kafka トピックにバインドし、Kafka シンクとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

43.3.2.2. クラスター CLI の使用手順

1. **postgresql-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。

2. 以下のコマンドを使用してシンクを実行します。

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

43.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

43.4. KAMELET ソースファイル

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

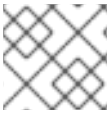
第44章 述語フィルターの動作

JsonPath 式に基づくフィルター

44.1. 設定オプション

以下の表では、**predicate-filter-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
expression *	式	外部括弧なしで評価する JsonPath 式。これは、例の foo フィールドが John に等しい場合、メッセージは続行され、そうでない場合はフィルターリングされることを意味しています。	string		**"@.foo =~ /.*John/"**



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

44.2. DEPENDENCIES

実行時に **predicate-filter-action** Kamelet は、以下の依存関係の存在に依存します。

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

44.3. 使用方法

本セクションでは、**predicate-filter-action** を使用する方法を説明します。

44.3.1. Knative Action

predicate-filter-action Kamelet を Knative バインディングの中間ステップとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

44.3.1.2. クラスター CLI の使用手順

1. **predicate-filter-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

44.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

44.3.2. Kafka Action

predicate-filter-action Kamelet を Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

44.3.2.2. クラスター CLI の使用手順

1. **predicate-filter-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

44.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

44.4. KAMELET ソースファイル

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

第45章 PROTOBUF デシリアライズアクション

ペイロードを Protobuf にデシリアライズ

45.1. 設定オプション

次の表は、**protobuf-deserialize-action** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
スキーマ *	スキーマ	(単一行として) シリアライズ時に使用する Protobuf スキーマ	string		**"message Person { required string first = 1; required string last = 2; }"**

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

45.2. DEPENDENCIES

実行時には、**protobuf-deserialize-action** Kamelet は以下の依存関係の存在に依存します。

- 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

ここでは、**protobuf-deserialize-action** の使用方法について説明します。

45.3.1. Knative Action

protobuf-deserialize-action Kamelet を Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

45.3.1.2. クラスター CLI の使用手順

1. **protobuf-deserialize-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

45.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

45.3.2. Kafka Action

protobuf-deserialize-action Kamelet を Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

45.3.2.2. クラスター CLI の使用手順

1. **protobuf-deserialize-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

45.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

45.4. KAMELET ソースファイル

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

第46章 PROTOBUF SERIALIZE アクション

ペイロードを Protobuf にシリアライズする

46.1. 設定オプション

次の表は、**protobuf-serialize-action** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
スキーマ *	スキーマ	(単一行として) シリアライズ時に使用する Protobuf スキーマ	string		**"message Person { required string first = 1; required string last = 2; }"**

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

46.2. DEPENDENCIES

実行時に、**protobuf-serialize-action** Kamelet は以下の依存関係の存在に依存します。

- 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

ここでは、**protobuf-serialize-action** の使用方法について説明します。

46.3.1. Knative Action

protobuf-serialize-action Kamelet を Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

46.3.1.2. クラスター CLI の使用手順

1. **protobuf-serialize-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

46.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

46.3.2. Kafka Action

protobuf-serialize-action Kamelet を Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

46.3.2.2. クラスター CLI の使用手順

1. **protobuf-serialize-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

46.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

46.4. KAMELET ソースファイル

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

第47章 REGEX ルーターの動作

設定された正規表現と代替文字列を使用して宛先を更新します。

47.1. 設定オプション

以下の表は、**regex-router-action** Kamelet で利用可能な設定オプションの概要を示しています。

プロパティー	名前	説明	型	デフォルト	例
regex *	Regex	宛先の正規表現	string		
replacement *	replacement	マッチした場合の置換	string		

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

47.2. DEPENDENCIES

実行時に **regex-router-action** Kamelet は、以下の依存関係の存在に依存します。

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

47.3. 使用方法

本セクションでは、**regex-router-action** の使用方法を説明します。

47.3.1. Knative Action

regex-router-action Kamelet を Knative バインディングの中間ステップとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

47.3.1.2. クラスター CLI の使用手順

1. **regex-router-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

47.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

47.3.2. Kafka Action

regex-router-action Kamelet を Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

47.3.2.2. クラスター CLI の使用手順

1. **regex-router-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

47.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

47.4. KAMELET ソースファイル

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

第48章 フィールドアクションの置き換え

送信中のメッセージのフィールドを別のキーに置き換えます。

- 必須パラメーター 'renames' は、たとえば 'foo:bar,abc:xyz' のようにコロンで区切られたリネームのペアのコンマ区切りリストで、フィールドリネームマッピングを表します。
- オプションのパラメーター 'enabled' は、含めるべきフィールドを表します。指定された場合、指定されたフィールドのみが結果のメッセージに含まれます。
- オプションのパラメーター 'disabled' は、除外するフィールドを表します。指定された場合、リストされたフィールドは結果のメッセージから除外されます。これは 'enabled' パラメーターよりも優先されます。
- 'enabled' パラメーターのデフォルト値は 'all' なので、ペイロードのすべてのフィールドが含まれることになります。
- 'disabled' パラメーターのデフォルト値は 'none' なので、ペイロードのどのフィールドも除外されることはありません。

48.1. 設定オプション

以下の表では、**replace-field-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
renames *	名前変更	名前を変更する新しい値を持つフィールドのコンマ区切りリスト	string		**"foo:bar,c1:c2"**
disabled	Disabled	無効にするフィールドのコンマ区切りリスト	string	"none"	
enabled	有効	有効にするフィールドのコンマ区切りリスト	string	"all"	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

48.2. DEPENDENCIES

実行時に、**replace-field-action** Kamelet は以下の依存関係の存在に依存します。

- 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

このセクションでは、**replace-field-action** の使用方法について説明します。

48.3.1. Knative Action

replace-field-action Kamelet は、Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

48.3.1.2. クラスター CLI の使用手順

1. **replace-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

48.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

48.3.2. Kafka Action

replace-field-action Kamelet は、Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

48.3.2.2. クラスター CLI の使用手順

1. **replace-field-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

48.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

48.4. KAMELET ソースファイル

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

第49章 SALESFORCE ソース

Salesforce から更新を受け取ります。

49.1. 設定オプション

次の表は、**salesforce-source** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
clientId *	Consumer Key	Salesforce アプリケーションコンシューマーキー	string		
clientSecret *	コンシューマーシークレット	Salesforce アプリケーションコンシューマーシークレット	string		
Password *	Password	Salesforce ユーザーのパスワード	string		
query *	クエリー	Salesforce で実行するクエリー	string		**"SELECT Id, Name, Email, Phone FROM Contact"**
topicName *	トピック名	使用するトピック/チャネルの名前	string		**"ContactTopic"**
userName *	Username	Salesforce のユーザー名	string		
loginUrl	ログイン URL	Salesforce インスタンスのログイン URL	string	**"https://login.salesforce.com"**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

49.2. DEPENDENCIES

salesforce-source Kamelet は、実行時に以下の依存関係の存在に依存します。

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

- camel:kamelet

49.3. 使用方法

ここでは、**salesforce-source** の使用方法について説明します。

49.3.1. Knative ソース

salesforce-source Kamelet を Knative オブジェクトにバインドすることで、Knative ソースとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

49.3.1.2. クラスター CLI の使用手順

1. **salesforce-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

49.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

49.3.2. Kafka Source

salesforce-source Kamelet を Kafka トピックにバインドすることにより、Kafka ソースとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

49.3.2.2. クラスター CLI の使用手順

1. **salesforce-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

49.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

49.4. KAMELET ソースファイル

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

第50章 SALESFORCE の作成シンク

Salesforce にオブジェクトを作成します。メッセージの本文には、salesforce オブジェクトの JSON を含める必要があります。

Example body: { "Phone": "555", "Name": "Antonia", "LastName": "Garcia" }

50.1. 設定オプション

次の表は、**salesforce-create-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
clientId *	Consumer Key	Salesforce アプリケーションコンシューマーキー	string		
clientSecret *	コンシューマーシークレット	Salesforce アプリケーションコンシューマーシークレット	string		
Password *	Password	Salesforce ユーザーのパスワード	string		
userName *	Username	Salesforce のユーザー名	string		
loginUrl	ログイン URL	Salesforce インスタンスのログイン URL	string	**"https://login.salesforce.com"**	
sObjectName	オブジェクト名	オブジェクトのタイプ	string		**"Contact"**

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

50.2. DEPENDENCIES

salesforce-create-sink Kamelet は、実行時に以下の依存関係の存在に依存します。

- camel:salesforce
- camel:kamelet

50.3. 使用方法

ここでは、**salesforce-create-sink** の使用方法について説明します。

50.3.1. Knative Sink

salesforce-create-sink Kamelet を Knative オブジェクトにバインドすることで、Knative シンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

50.3.1.2. クラスター CLI の使用手順

1. **salesforce-create-sink-binding.yaml** ファイルをローカルドライブに保存してから、設定に合わせて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

50.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

50.3.2. Kafka Sink

salesforce-create-sink Kamelet を Kafka トピックにバインドすることで、Kafka シンクとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

50.3.2.2. クラスター CLI の使用手順

1. **salesforce-create-sink-binding.yaml** ファイルをローカルドライブに保存してから、設定に合わせて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

50.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

50.4. KAMELET ソースファイル

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

第51章 SALESFORCE の削除シンク

Salesforce からオブジェクトを削除します。受信する本文は、sObjectId と sObjectName の 2 つのキーを含む JSON である必要があります。

Example body: { "sObjectId": "XXXXX0", "sObjectName": "Contact" }

51.1. 設定オプション

次の表は、**salesforce-delete-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
clientId *	Consumer Key	Salesforce アプリケーションコンシューマーキー	string		
clientSecret *	コンシューマーシークレット	Salesforce アプリケーションコンシューマーシークレット	string		
Password *	Password	Salesforce ユーザーのパスワード	string		
userName *	Username	Salesforce のユーザー名	string		
loginUrl	ログイン URL	Salesforce インスタンスのログイン URL	string	**"https://login.salesforce.com"**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

51.2. DEPENDENCIES

salesforce-delete-sink Kamelet は、実行時に以下の依存関係の存在に依存します。

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

51.3. 使用方法

このセクションでは、**salesforce-delete-sink** の使用方法について説明します。

51.3.1. Knative Sink

salesforce-delete-sink Kamelet を Knative オブジェクトにバインドすることで、Knative シンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

51.3.1.2. クラスター CLI の使用手順

1. **salesforce-delete-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

51.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

51.3.2. Kafka Sink

salesforce-delete-sink Kamelet を Kafka トピックにバインドすることで、Kafka シンクとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

51.3.2.2. クラスター CLI の使用手順

1. **salesforce-delete-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

51.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

51.4. KAMELET ソースファイル

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

第52章 SALESFORCE の更新シンク

Salesforce のオブジェクトを更新します。受信した本文には、更新する各プロパティーの JSON キーと値のペアが含まれている必要があり、sObjectName と sObjectId をパラメーターとして指定する必要があります。

キーと値のペアの例: { "Phone": "1234567890", "Name": "Antonia" }

52.1. 設定オプション

次の表は、**salesforce-update-sink** Kamelet で利用できる設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
clientId *	Consumer Key	Salesforce アプリケーションコンシューマーキー	string		
clientSecret *	コンシューマーシークレット	Salesforce アプリケーションコンシューマーシークレット	string		
Password *	Password	Salesforce ユーザーのパスワード	string		
sObjectId *	オブジェクト ID	オブジェクトの ID。キーと値のペアを使用する場合にのみ必要です。	string		
sObjectName *	オブジェクト名	オブジェクトのタイプ。キーと値のペアを使用する場合にのみ必要です。	string		**"Contact"**
userName *	Username	Salesforce のユーザー名	string		
loginUrl	ログイン URL	Salesforce インスタンスのログイン URL	string	**"https://login.salesforce.com"**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

52.2. DEPENDENCIES

salesforce-update-sink Kamelet は、実行時に以下の依存関係の存在に依存します。

- camel:salesforce
- camel:kamelet

52.3. 使用方法

ここでは、**salesforce-update-sink** の使用方法について説明します。

52.3.1. Knative Sink

salesforce-update-sink Kamelet を Knative オブジェクトにバインドすることで、Knative シンクとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

52.3.1.2. クラスター CLI の使用手順

1. **salesforce-update-sink-binding.yaml** ファイルをローカルドライブに保存してから、設定に合わせて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

52.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

52.3.2. Kafka Sink

salesforce-update-sink Kamelet を Kafka トピックにバインドすることで、Kafka シンクとして使用することが可能です。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

52.3.2.2. クラスター CLI の使用手順

1. **salesforce-update-sink-binding.yaml** ファイルをローカルドライブに保存してから、設定に合わせて編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

52.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

52.4. KAMELET ソースファイル

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

第53章 SFTP SINK

SFTP サーバーにデータを送信します。

Kamelet では、以下のヘッダーが設定されていることを想定しています。

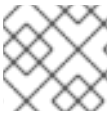
- **file/ce-file**: アップロードするファイル名として

ヘッダーが設定されていない場合、エクスチェンジ ID はファイル名として使用されます。

53.1. 設定オプション

以下の表では、**sftp-sink** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
connectionHost *	接続ホスト	FTP サーバーのホスト名	string		
connectionPort *	接続ポート	FTP サーバーのポート	string	**22**	
directoryName *	ディレクトリー名	開始ディレクトリー	string		
Password *	Password	FTP サーバーにアクセスするためのパスワード	string		
username *	Username	FTP サーバーにアクセスするためのユーザー名	string		
fileExist	ファイルの存在	すでにファイルが存在する場合にどのように動作するか。列挙は 4 つあり、値は Override、Append、Fail または Ignore のいずれかです。	string	**"Override"**	
passiveMode	パッシブモード	パッシブモード接続の設定	boolean	**false**	



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

53.2. DEPENDENCIES

ランタイム時に、**sftp-sink** Kamelet は、以下の依存関係の存在に依存します。

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

53.3. 使用方法

本セクションでは、**sftp-sink** を使用する方法を説明します。

53.3.1. Knative Sink

sftp-sink Kamelet を Knative オブジェクトにバインドすることにより、これを Knative シンクとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

53.3.1.2. クラスター CLI の使用手順

1. **sftp-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

53.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

53.3.2. Kafka Sink

sftp-sink Kamelet を Kafka トピックにバインドすることにより、Kafka シンクとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

53.3.2.2. クラスター CLI の使用手順

1. **sftp-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。
2. 以下のコマンドを使用してシンクを実行します。

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

53.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

53.4. KAMELET ソースファイル

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

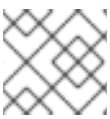
第54章 SFTP ソース

SFTP サーバーからデータを受け取ります。

54.1. 設定オプション

以下の表では、**sftp-source** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
connectionHost *	接続ホスト	SFTP サーバーのホスト名	string		
connectionPort *	接続ポート	FTP サーバーのポート	string	**22**	
directoryName *	ディレクトリー名	開始ディレクトリー	string		
Password *	Password	SFTP サーバーにアクセスするためのパスワード	string		
username *	Username	SFTP サーバーにアクセスするためのユーザー名	string		
idempotent	冪等性	処理されたファイルを省略します。	boolean	**true**	
passiveMode	パッシブモード	パッシブモード接続の設定	boolean	**false**	
再帰	再帰	ディレクトリーの場合は、すべてのサブディレクトリー内のファイルも検索します。	boolean	**false**	



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

54.2. DEPENDENCIES

ランタイム時に、**sftp-source** Kamelet は、以下の依存関係の存在に依存します。

- camel:ftp

- camel:core
- camel:kamelet

54.3. 使用方法

本セクションでは、**sftp-source** を使用する方法を説明します。

54.3.1. Knative ソース

sftp-source Kamelet を Knative オブジェクトにバインドすることにより、これを Knative ソースとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

54.3.1.2. クラスター CLI の使用手順

1. **sftp-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

54.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

54.3.2. Kafka Source

sftp-source Kamelet を Kafka トピックにバインドすることにより、Kafka ソースとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

54.3.2.2. クラスター CLI の使用手順

1. **sftp-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

54.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
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
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

54.4. KAMELET ソースファイル

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

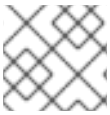
第55章 SLACK ソース

Slack チャネルからメッセージを受信します。

55.1. 設定オプション

以下の表では、**slack-source** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
channel *	チャネル	メッセージを受信する Slack チャネル	string		**"#myroom"**
token *	トークン	Slack にアクセスするためのトークン。Slack アプリが必要です。このアプリには、channels:history と channels:read の権限が必要です。Bot User OAuth Access Token は、必要な種類のトークンです。	string		



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

55.2. DEPENDENCIES

ランタイム時に、**slack-source** Kamelet は、以下の依存関係の存在に依存します。

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

55.3. 使用方法

本セクションでは、**slack-source** を使用する方法を説明します。

55.3.1. Knative ソース

slack-source Kamelet を Knative オブジェクトにバインドすることにより、これを Knative ソースとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

55.3.1.2. クラスター CLI の使用手順

1. **slack-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

55.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

55.3.2. Kafka Source

slack-source Kamelet を Kafka トピックにバインドすることにより、Kafka ソースとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

55.3.2.2. クラスター CLI の使用手順

1. **slack-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

55.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

55.4. KAMELET ソースファイル

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

第56章 MICROSOFT SQL SERVER SINK

Microsoft SQL Server データベースにデータを送信します。

この Kamelet は JSON をボディーとして想定します。JSON フィールドとパラメーター間のマッピングはキーで実行されるため、以下のクエリーがある場合は以下を実行します。

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

Kamelet は、以下のような入力として受信する必要があります。

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

56.1. 設定オプション

以下の表は、**sqlserver-sink** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
databaseName *	データベース名	ポイントするデータベース名	string		
Password *	Password	セキュアな SQL Server データベースへのアクセスに使用するパスワード	string		
query *	クエリー	SQL Server データベースに対して実行するクエリー	string		**"INSERT INTO accounts (username,city) VALUES (:#username,:#city)"**
serverName *	サーバー名	データソースのサーバー名	string		**"localhost"**
username *	Username	セキュアな SQL Server データベースへのアクセスに使用するユーザー名	string		
serverPort	サーバーポート	データソースのサーバーポート	string	**1433**	

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

56.2. DEPENDENCIES

ランタイム時に、**sqlserver-sink** Kamelet は、以下の依存関係の存在に依存します。

- 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

本セクションでは、**sqlserver-sink** を使用する方法を説明します。

56.3.1. Knative Sink

sqlserver-sink Kamelet を Knative オブジェクトにバインドすることにより、Knative シンクとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

56.3.1.2. クラスター CLI の使用手順

1. **sqlserver-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。

2. 以下のコマンドを使用してシンクを実行します。

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

56.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

56.3.2. Kafka Sink

sqlserver-sink Kamelet を Kafka シンクとして使用することは、Kafka トピックにバインドします。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

56.3.2.2. クラスター CLI の使用手順

1. **sqlserver-sink-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせてこれを編集します。

2. 以下のコマンドを使用してシンクを実行します。

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

56.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してシンクを設定し、実行します。

```
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"
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

56.4. KAMELET ソースファイル

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

第57章 TELEGRAM ソース

Telegram ボットに送信されたすべてのメッセージを受信します。

ボットを作成するには、Telegram アプリケーションを使用して @botfather アカウントにお問い合わせください。

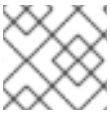
ソースは、以下のヘッダーをメッセージに割り当てます。

- **chat-id** / **ce-chatid**: メッセージが出るチャットの ID

57.1. 設定オプション

以下の表に、**telegram-source** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
authorizationToken *	トークン	Telegram でボットにアクセスするためのトークン。Telegram @botfather から取得できます。	string		



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

57.2. DEPENDENCIES

実行時には、**telegram-source** Kamelet は以下の依存関係の存在に依存します。

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

57.3. 使用方法

本セクションでは、**telegram-source** の使用方法を説明します。

57.3.1. Knative ソース

telegram-source Kamelet を Knative オブジェクトにバインドすることにより、Knative ソースとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

57.3.1.2. クラスター CLI の使用手順

1. **telegram-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

57.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

57.3.2. Kafka Source

telegram-source Kamelet を Kafka トピックにバインドすることにより、Kafka のソースとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

57.3.2.2. クラスター CLI の使用手順

1. **telegram-source-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用してソースを実行します。

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

57.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
kamel bind telegram-source -p "source.authorizationToken=The Token"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

57.4. KAMELET ソースファイル

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

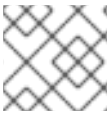
第58章 スロットルアクション

スロットルアクションを使用すると、特定のシンクが過負荷にならないようにすることができます。

58.1. 設定オプション

以下の表では、**throttle-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
messages *	Messages Number	設定した時間内に送信するメッセージ数	integer		**10**
timePeriod	Time Period	最大要求数の有効期限をミリ秒単位で設定する	string	**"1000"**	



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

58.2. DEPENDENCIES

実行時に、**throttle-action** Kamelet は以下の依存関係の存在に依存します。

- camel:core
- camel:kamelet

58.3. 使用方法

ここでは、**throttle-action** の使用方法について説明します。

58.3.1. Knative Action

throttle-action Kamelet は、Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

58.3.1.2. クラスター CLI の使用手順

1. **throttle-action-binding.yaml** ファイルをローカルドライブに保存してから、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

```
oc apply -f throttle-action-binding.yaml
```

58.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
kamel bind timer-source?message=Hello --step throttle-action -p "step-0.messages=10"
channel:mychannel
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

58.3.2. Kafka Action

throttle-action Kamelet は、Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

58.3.2.2. クラスター CLI の使用手順

1. **throttle-action-binding.yaml** ファイルをローカルドライブに保存してから、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

```
oc apply -f throttle-action-binding.yaml
```

58.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
kamel bind timer-source?message=Hello --step throttle-action -p "step-0.messages=1"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

58.4. KAMELET ソースファイル

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

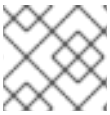
第59章 タイマーソース

カスタムペイロードを使用して定期的なイベントを生成します。

59.1. 設定オプション

次の表は、**timer-source** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
message *	メッセージ	生成するメッセージ	string		**"hello world"**
contentType	コンテンツタイプ	生成されるメッセージのコンテンツタイプ	string	**"text/plain"**	
period	期間	2 つのイベントの間隔 (ミリ秒単位)	integer	**1000**	



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

59.2. DEPENDENCIES

timer-source Kamelet は、実行時に以下の依存関係の存在に依存します。

- camel:core
- camel:timer
- camel:kamelet

59.3. 使用方法

ここでは、**timer-source** の使用方法について説明します。

59.3.1. Knative ソース

timer-source Kamelet を Knative オブジェクトにバインドすることにより、Knative ソースとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

59.3.1.2. クラスター CLI の使用手順

1. **timer-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してソースを実行します。

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

59.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
kamel bind timer-source -p "source.message=hello world" channel:mychannel
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

59.3.2. Kafka Source

timer-source Kamelet を Kafka のトピックにバインドすることで、Kafka のソースとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

59.3.2.2. クラスター CLI の使用手順

1. **timer-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集してください。
2. 以下のコマンドを使用してソースを実行します。

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

59.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用してソースを設定および実行します。

```
kamel bind timer-source -p "source.message=hello world" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

59.4. KAMELET ソースファイル

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

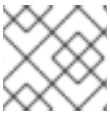
第60章 ルーターのタイムスタンプアクション

topic フィールドを、元のトピック名およびレコードのタイムスタンプとして更新します。

60.1. 設定オプション

次の表は、**timestamp-router-action** Kamelet で利用可能な設定オプションをまとめたものです。

プロパティー	名前	説明	型	デフォルト	例
timestampFormat	タイムスタンプの形式	java.text.SimpleDateFormat と互換性があるタイムスタンプの文字列の文字列。	string	**"yyyyMMdd"**	
timestampHeaderName	タイムスタンプヘッダー名	タイムスタンプが含まれるヘッダーの名前	string	**"kafka.TIMESTAMP"**	
topicFormat	トピックの形式	それぞれトピックとタイムスタンプのプレースホルダーとして '$[topic]' および '$[timestamp]' を含む文字列のフォーマット文字列。	string	**"topic-$[timestamp]"**	



注記

アスタリスク (*) のマークが付いたフィールドは必須です。

60.2. DEPENDENCIES

実行時、**timestamp-router-action** Kamelet は以下の依存関係の存在に依存しています。

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

60.3. 使用方法

ここでは、**timestamp-router-action** の使用方法について説明します。

60.3.1. Knative Action

timestamp-router-action Kamelet は、Knative バインディングの中間ステップとして使用することができます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

60.3.1.2. クラスター CLI の使用手順

1. **timestamp-router-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

60.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
kamel bind timer-source?message=Hello --step timestamp-router-action channel:mychannel
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

60.3.2. Kafka Action

timestamp-router-action Kamelet は、Kafka バインディングの中間ステップとして使用することができます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

60.3.2.2. クラスター CLI の使用手順

1. **timestamp-router-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて設定を編集します。
2. 以下のコマンドを使用して、アクションを実行します。

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

60.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

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

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

60.4. KAMELET ソースファイル

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

第61章 キー動作に対する値

Kafka レコードキーを、ボディーのフィールドのサブセットから形成された新しいキーに置き換えます。

61.1. 設定オプション

以下の表には、**value-to-key-action** Kamelet で利用可能な設定オプションをまとめています。

プロパティー	名前	説明	型	デフォルト	例
fields *	フィールド	新しいキーを設定するために使用されるフィールドのコンマ区切りリスト	string		

注記

アスタリスク (*) のマークが付いたフィールドは必須です。

61.2. DEPENDENCIES

ランタイム時に、**value-to-key-action** Kamelet は、以下の依存関係の存在に依存します。

- 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

本セクションでは、**value-to-key-action** を使用する方法を説明します。

61.3.1. Knative Action

value-to-key-action Kamelet を Knative バインディングの中間ステップとして使用できます。

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

接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

61.3.1.2. クラスター CLI の使用手順

1. **value-to-key-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

```
oc apply -f value-to-key-action-binding.yaml
```

61.3.1.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
kamel bind timer-source?message=Hello --step value-to-key-action -p "step-0.fields=The Fields"
channel:mychannel
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

61.3.2. Kafka Action

value-to-key-action Kamelet を Kafka バインディングの中間ステップとして使用できます。

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. 前提条件

AMQ Streams Operator を OpenShift クラスターにインストールし、現在の namespace に **my-topic** という名前のトピックを作成していることを確認します。接続先の OpenShift クラスターに**Red Hat Integration - Camel K**がインストールされていることを確認します。

61.3.2.2. クラスター CLI の使用手順

1. **value-to-key-action-binding.yaml** ファイルをローカルドライブに保存し、設定に合わせて編集します。
2. 以下のコマンドを使用して、アクションを実行します。

```
oc apply -f value-to-key-action-binding.yaml
```

61.3.2.3. Kamel CLI を使用するための手順

以下のコマンドを使用して、アクションを設定および実行します。

```
kamel bind timer-source?message=Hello --step value-to-key-action -p "step-0.fields=The Fields"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

このコマンドは、クラスターの現在の namespace に KameletBinding を作成します。

61.4. KAMELET ソースファイル

https://github.com/openshift-integration/kamelet-catalog/tree/kamelet-catalog-1.8//value-to-key-action.kamelet.yaml