Red Hat Integration 2022.Q1

Kamelets リファレンス

Kamelets リファレンス

Last Updated: 2023-03-25

Red Hat Integration 2022.Q1 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 サポート対象の KAMELETS

以下の Kamelets は Camel K 1.6 でサポートされます。

- Avro デシリアライズアクション
- Avro シリアライズアクション
- AWS 2 S3 シンク
- AWS 2 S3 ソース
- AWS 2 S3 streaming upload sink
- AWS 2 Kinesis シンク
- AWS 2 Kinesis ソース
- AWS 2 Lambda シンク
- AWS 2 Simple Notification System シンク
- AWS 2 Simple Queue Service シンク
- AWS 2 Simple Queue Service ソース
- AWS SQS FIFO シンク
- Cassandra シンク （テクノロジープレビュー）
- Cassandra ソース （テクノロジープレビュー）
- Elasticsearch インデックスシンク （テクノロジープレビュー）
- フィールドアクションの抽出
- FTP シンク
- FTP ソース
- Header Filter アクションがある
- Hoist フィールドアクション
- HTTP シンク
- フィールドアクションの挿入
- Header アクションの挿入
- Tombstone Filter アクション
- JIRA ソース （テクノロジープレビュー）
- JMS シンク
- JMS ソース

- JSON デシリアライズアクション
- JSON シリアライズアクション
- Kafka シンクコ
- Kafka ソース
- Kafka Topic name filter action （Kafka のみ）
- ログシンク （開発およびテストの目的で）
- MariaDB シンク
- mask フィールドアクション
- Message TimeStamp アクション
- MongoDB シンク
- MongoDB ソース
- MySQL シンクコネクター
- PostgreSQL シンクコネクター
- 述語フィルターアクション
- Protobuf デシリアライズアクション
- Protobuf シリアライズアクション
- regex ルーターのアクション
- フィールドアクションの置き換え
- Salesforce ソース
- SFTP シンク
- SFTP ソース
- Slack ソース
- SQL Server シンク
- Telegram ソース （テクノロジープレビュー）
- タイマーソース （開発およびテスト目的のため）
- ルーターアクションのタイムスタンプ
- キーアクションに対する値

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	Name (名前)	詳細	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。	ブール値	**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. 依存関係

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。

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

2.3. 用途

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 "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-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。

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 "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-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。

2.4. KAMELET ソースファイル

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/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	Name (名前)	詳細	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。	ブール値	**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. 依存関係

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。

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

3.3. 用途

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 "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 "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/blob/kamelet-catalog-1.6/avro-serialize-action.kamelet.yaml

第4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。

4.1. 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	Name (名前)	詳細	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。	ブール値	**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。

4.2. 依存関係

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

4.3. 用途

ここでは、**aws-s3-sink**の使用方法について説明します。

4.3.1. 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"
```

4.3.1.1. 前提条件

接続先のOpenShiftクラスタに**「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-s3-sink-binding.yaml** ファイルをローカルドライブに保存してから、構成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
```

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-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。

4.3.2. 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"
```

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-s3-sink-binding.yaml** ファイルをローカルドライブに保存してから、構成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
```

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-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。

4.4. KAMELET ソースファイル

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

第5章 AWS S3 ソース

AWS S3 からデータを受け取ります。

5.1. 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	Name (名前)	詳細	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。	ブール値	**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	ブール値	**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。

5.2. 依存関係

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

5.3. 用途

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

5.3.1. Knative Source

aws-s3-source KameletをKnativeオブジェクトにバインドすることで、Knative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: 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-s3-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成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
```

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-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。

5.3.2. 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
```

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-s3-source-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成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
```

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-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。

5.4. KAMELET ソースファイル

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

第6章 AWS S3 STREAMING UPLOAD SINK

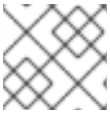
データをストリーミングアップロードモードで AWS S3 にアップロードします。

6.1. 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	Name (名前)	詳細	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。	ブール値	**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	Name (名前)	詳細	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	ブール値	**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。

6.2. 依存関係

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

6.3. 用途

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

6.3.1. 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"
```

6.3.1.1. 前提条件

接続先のOpenShiftクラスタに**「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-s3-streaming-upload-sink-binding.yaml**ファイルをローカルドライブに保存し、構成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
```

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-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。

6.3.2. 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"
```

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-s3-streaming-upload-sink-binding.yaml**ファイルをローカルドライブに保存し、構成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
```

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-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。

6.4. KAMELET ソースファイル

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

第7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。

7.1. 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	Name (名前)	詳細	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。

7.2. 依存関係

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

7.3. 用途

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

7.3.1. 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"
```

7.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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-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
```

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-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。

7.3.2. 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"
```

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-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
```

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-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。

7.4. KAMELET ソースファイル

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

第8章 AWS KINESIS ソース

AWS Kinesis からデータを受信します。

8.1. 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	Name (名前)	詳細	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。

8.2. 依存関係

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

8.3. 用途

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

8.3.1. Knative Source

aws-kinesis-source KameletをKnativeオブジェクトにバインドすることで、Knativeの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
```

8.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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-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ソースを実行します。

```
oc apply -f aws-kinesis-source-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。

8.3.2. 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
```

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-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ソースを実行します。

```
oc apply -f aws-kinesis-source-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。

8.4. KAMELET ソースファイル

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

第9章 AWS LAMBDA SINK

AWS Lambda 関数へのペイロードの送信

9.1. 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	Name (名前)	詳細	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。

9.2. 依存関係

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

9.3. 用途

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

9.3.1. 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"
```

9.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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-lambda-sink-binding.yaml**ファイルをローカルドライブに保存してから、構成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
```

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-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。

9.3.2. 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"
```

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-lambda-sink-binding.yaml**ファイルをローカルドライブに保存してから、構成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
```

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-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。

9.4. KAMELET ソースファイル

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

第10章 AWS SNS SINK

AWS SNS トピックへのメッセージの送信

10.1. 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	Name (名前)	詳細	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。	ブール値	**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。

10.2. 依存関係

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

10.3. 用途

ここでは、**aws-s3-sink**の使用方法について説明します。

10.3.1. 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"
```

10.3.1.1. 前提条件

接続先のOpenShiftクラスタに**「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-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
```

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。

10.3.2. 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"
```

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-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
```

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。

10.4. KAMELET ソースファイル

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

第11章 AWS SQS SINK

メッセージを AWS SQS キューに送信します。

11.1. 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	Name (名前)	詳細	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。	ブール値	**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. 依存関係

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

11.3. 用途

ここでは、**aws-sqs-sink** の使用方法について説明します。

11.3.1. 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"
```

11.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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-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-sqs-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-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-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"
```

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-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-sqs-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-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/blob/kamelet-catalog-1.6/aws-sqs-sink.kamelet.yaml

第12章 AWS SQS ソース

AWS SQS からデータを受け取ります。

12.1. 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	Name (名前)	詳細	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。	ブール値	**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	ブール値	**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。

12.2. 依存関係

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

12.3. 用途

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

12.3.1. Knative Source

aws-sqs-source KameletをKnativeオブジェクトにバインドすることで、Knative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: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

12.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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-sqs-source-binding.yaml** ファイルをローカルドライブに保存し、構成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
```

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。

12.3.2. 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
```

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-sqs-source-binding.yaml** ファイルをローカルドライブに保存し、構成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
```

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。

12.4. KAMELET ソースファイル

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

第13章 AWS SQS FIFO SINK

AWS SQS FIFO キューへのメッセージの送信

13.1. 設定オプション

以下の表は、**aws-sqs-fifo-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	Name (名前)	詳細	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。	ブール値	**false**	
contentBasedDeduplication	Content-Based Deduplication	コンテンツベースの重複排除を使用します（最初に SQS FIFO キューで有効化されるはずです）	ブール値	**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。

13.2. 依存関係

ランタイム時に、**aws-sqs-fifo-sink** Kamelet は以下の依存関係の有無に依存します。

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

13.3. 用途

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

13.3.1. Knative Sink

aws-sqs-fifo-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。

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"
```

13.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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-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
```

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。

13.3.2. Kafka Sink

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

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"
```

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-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
```

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。

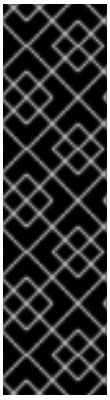
```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic aws-sqs-fifo-sink -p "sink.accessKey=The Access Key" -p "sink.queueNameOrArn=The Queue Name" -p "sink.region=eu-west-1" -p "sink.secretKey=The Secret Key"
```

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。

13.4. KAMELET ソースファイル

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

第14章 CASSANDRA SINK



重要

Cassandra Sink Kamelet はテクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。

これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、「テクノロジプレビュー機能のサポート範囲」を参照してください。

データを Cassandra クラスターに送信します。

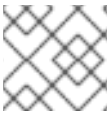
この Kamelet はボディーを JSON アレイとして想定します。JSON アレイの内容は、クエリーパラメーターに設定された CQL Prepared Statement の入力として使用されます。

14.1. 設定オプション

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

プロパティー	Name (名前)	詳細	型	デフォルト	例
connectionHost *	接続ホスト	Hostname(s) cassandra server(s). 複数のホストを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 *	パスワード	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		
preparedStatement *	準備済みステートメント	Cassandra クラスターテーブルに対して実行する準備済みステートメント	string		

プロパティー	Name (名前)	詳細	型	デフォルト	例
username *	ユーザ名	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"**	



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. 依存関係

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

14.3. 用途

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

14.3.1. 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"
      preparedStatement: "The Prepared Statement"
      username: "The Username"
```

14.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **cassandra-sink-binding.yaml** ファイルをローカル・ドライブに保存し、構成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
```

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 cassandra-sink -p "sink.connectionHost=localhost" -p
sink.connectionPort=9042 -p "sink.keyspace=customers" -p "sink.password=The Password" -p
"sink.preparedStatement=The Prepared Statement" -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。

14.3.2. 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"
    preparedStatement: "The Prepared Statement"
    username: "The Username"
```

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. **cassandra-sink-binding.yaml** ファイルをローカル・ドライブに保存し、構成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
```

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。

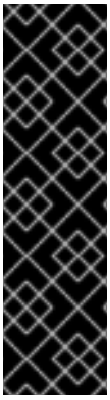
```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic cassandra-sink -p
"sink.connectionHost=localhost" -p sink.connectionPort=9042 -p "sink.keyspace=customers" -p
"sink.password=The Password" -p "sink.preparedStatement=The Prepared Statement" -p
"sink.username=The Username"
```

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。

14.4. KAMELET ソースファイル

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

第15章 CASSANDRA ソース



重要

Cassandra Source Kamelet はテクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。

これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、「テクノロジプレビュー機能のサポート範囲」を参照してください。

Cassandra クラスターテーブルをクエリーします。

15.1. 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	Name (名前)	詳細	型	デフォルト	例
connectionHost *	接続ホスト	Hostname(s) cassandra server(s). 複数のホストを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 *	パスワード	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 *	ユーザ名	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		

プロパティー	Name (名前)	詳細	型	デフォルト	例
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"**	
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。

15.2. 依存関係

起動時に、**cassandra-source** Kamelet は、以下の依存関係の有無に依存します。

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

15.3. 用途

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

15.3.1. Knative Source

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
```

15.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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-source-binding.yaml**ファイルをローカル・ドライブに保存し、必要に応じて構成を編集します。
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
```

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。

15.3.2. 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
```

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-source-binding.yaml**ファイルをローカル・ドライブに保存し、必要に応じて構成を編集します。
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
```

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。

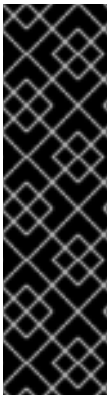
```
kamel bind cassandra-source -p "source.connectionHost=localhost" -p source.connectionPort=9042 -p "source.keyspace=customers" -p "source.password=The Password" -p "source.query=The Query" -p "source.username=The Username" kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

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。

15.4. KAMELET ソースファイル

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

第16章 ELASTICSEARCH INDEX SINK



重要

ElasticSearch Index Kameletは、技術プレビューの機能のみです。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。

これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、「テクノロジプレビュー機能のサポート範囲」を参照してください。

このシンクには、ドキュメントが ElasticSearch に保存されます。

入力データには、使用されるインデックスに応じて JSON 形式が必要です。

Kamelet には以下のヘッダーが必要です。

- **indexId/ce-indexid**: Elasticsearchのインデックスとして。

ヘッダーが設定されていない場合、インデックスは ES クラスターによって生成されます。

- **indexName/ce-indexname**: Elasticsearchのインデックス名として使用します。

このヘッダーが設定されていない場合は、**camel-k-index-es** がインデックス名として使用されます。

16.1. 設定オプション

次の表は、**elasticsearch-index-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	Name (名前)	詳細	型	デフォルト	例
clusterName *	ElasticSearch Cluster Name	クラスターの名前。	string		**"quickstart"**
hostAddresses *	ホストアドレス	使用する ip:port 形式のリモートトランスポートアドレスのコンマ区切りリスト。	string		**"quickstart-es-http:9200"**
enableSSL	SSL の有効化	SSL を使用して接続しますか？	ブール値	**true**	
indexName	ElasticSearch のインデックス	動作させるインデックスの名前。	string		**"data"**

プロパティー	Name (名前)	詳細	型	デフォルト	例
password	パスワード	ElasticSearchに接続するためのパスワードです。	string		
user	ユーザ名	ElasticSearch に接続するためのユーザー名。	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。

16.2. 依存関係

実行時に、**elasticsearch-index-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
- mvn:org.apache.camel.k:camel-k-kamelet-reify
- camel:elasticsearch-rest
- camel:gson
- camel:bean

16.3. 用途

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

16.3.1. Knative Sink

elasticsearch-index-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。

elasticsearch-index-sink-binding.yaml

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

```
    apiVersion: messaging.knative.dev/v1
    name: mychannel
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: elasticsearch-index-sink
    properties:
      clusterName: "quickstart"
      hostAddresses: "quickstart-es-http:9200"
```

16.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **elasticsearch-index-sink-binding.yaml**ファイルをローカルドライブに保存し、必要に応じて構成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 elasticsearch-index-sink-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シンクを設定し、実行します。

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

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 Sink

elasticsearch-index-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。

elasticsearch-index-sink-binding.yaml

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

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

16.3.2.1. 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. **elasticsearch-index-sink-binding.yaml**ファイルをローカルドライブに保存し、必要に応じて構成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 elasticsearch-index-sink-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シンクを設定し、実行します。

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

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/blob/kamelet-catalog-1.6/elasticsearch-index-sink.kamelet.yaml

第17章 フィールドアクションの抽出

ボディーからフィールドを抽出

17.1. 設定オプション

以下の表は、**extract-field-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	Name (名前)	詳細	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. 依存関係

ランタイム時に、**extract-field-action** 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:1.0.0.fuse-800048-redhat-00001
- camel:kamelet
- camel:core
- camel:jackson

17.3. 用途

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

17.3.1. Knative Action

extract-field-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。

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、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 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 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。

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、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 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/blob/kamelet-catalog-1.6/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	Name (名前)	詳細	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	起動ディレクトリー	string		
Password *	パスワード	FTPサーバーにアクセスするためのパスワード	string		
username *	ユーザ名	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	ブール値	**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. 依存関係

起動時に、**ftp-sink** Kamelet は、以下の依存関係の有無に依存します。

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

18.3. 用途

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/blob/kamelet-catalog-1.6/ftp-sink.kamelet.yaml

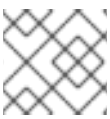
第19章 FTP ソース

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	Name (名前)	詳細	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	起動ディレクトリー	string		
Password *	パスワード	FTPサーバーにアクセスするためのパスワード	string		
username *	ユーザ名	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。	ブール値	**true**	
passiveMode	パッシブモード	パッシブモード接続の設定	ブール値	**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。	ブール値	**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. 依存関係

起動時に、**ftp-source** Kamelet は、以下の依存関係の有無に依存します。

- camel:ftp

- camel:core
- camel:kamelet

19.3. 用途

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 Source

ftp-source KameletをKnativeオブジェクトにバインドすることで、Knative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: 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、必要に応じて構成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、必要に応じて構成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/blob/kamelet-catalog-1.6/ftp-source.kamelet.yaml

第20章 HAS HEADER FILTER ACTION

1 つのヘッダーの有無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	Name (名前)	詳細	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 で渡す必要があります。Knative のみ、ヘッダーの名前には CloudEvent(**ce-**)プレフィックスが必要です。	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. 依存関係

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

- camel:core
- camel:kamelet

20.3. 用途

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。この例では、Knative **mychannel** は **ce-foo** という名前のメッセージヘッダーを提供します。ヘッダー名の CloudEvents(**ce-**)プレフィックスが必要です。Insert Header アクション の例では、メッセージヘッダーをデータに追加する方法を示します。

has-header-filter-action-binding.yaml

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

```
source:
  ref:
    kind: Channel
    apiVersion: messaging.knative.dev/v1
    name: mychannel
steps:
- ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: has-header-filter-action
  properties:
    name: "ce-foo"
sink:
  ref:
    kind: Kamelet
    apiVersion: camel.apache.org/v1alpha1
    name: log-sink
```

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てください。
- Kamelet Binding のソース Kamelet は、**has-header-filter-action** Kamelet の名前プロパティーで指定した **名前** のヘッダーを渡す必要があります。

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 channel:mychannel --step has-header-filter-action -p "step-0.name=ce-foo" 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。

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。この例では、**kafka-source** Kamelet は **foo** という名前のメッセージヘッダーを提供します。Insert Header アクション の例では、メッセージヘッダーをデータに追加する方法を示します。

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: kafka-source
    properties:
      bootstrapServers: "my-cluster-kafka-bootstrap.myproject.svc:9092"
      password: "XXX"
      topic: "my-topic"
      user: "XXX"
      securityProtocol: "PLAINTEXT"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: has-header-filter-action
    properties:
      name: "foo"
  sink:
   ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: log-sink
```

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てください。

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 kafka-source -p "source.bootstrapServers=my-cluster-kafka-
bootstrap.myproject.svc:9092" -p "source.password=XXX" -p "source.topic=my-topic" -p
"source.user=XXX" -p "source.securityProtocol=PLAINTEXT" --step has-header-filter-action -p "step-
0.name=foo" 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。

20.4. KAMELET ソースファイル

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

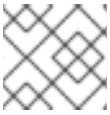
第21章 ハストフィールドアクション

単一のフィールドにデータをラップする

21.1. 設定オプション

以下の表は、**hoist-field-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	Name (名前)	詳細	型	デフォルト	例
field *	フィールド	イベントが含まれ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。

21.2. 依存関係

実行時に、**hoist-field-action** Kamelet は以下の依存関係の有無に依存します。

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

21.3. 用途

本セクションでは、ho **ist-field-action を** 使用する方法を説明します。

21.3.1. Knative Action

hoist-field-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。

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、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 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 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。

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、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 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/blob/kamelet-catalog-1.6/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	Name (名前)	詳細	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"**
method	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. 依存関係

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. 用途

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/blob/kamelet-catalog-1.6/http-sink.kamelet.yaml

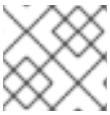
第23章 フィールドアクションの挿入

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	Name (名前)	詳細	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. 依存関係

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。

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

23.3. 用途

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: "Hello"
steps:
- 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、必要に応じて構成を編集します。
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 timer-source?message=Hello --step insert-field-action -p "step-0.field=The Field" -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。

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: "Hello"
  steps:
  - 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、必要に応じて構成を編集します。
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 timer-source?message=Hello --step insert-field-action -p "step-0.field=The Field" -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。

23.4. KAMELET ソースファイル

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

第24章 ヘッダーアクションの挿入

定数値を持つヘッダーを転送のメッセージに追加します。

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	Name (名前)	詳細	型	デフォルト	例
name *	Name (名前)	追加するヘッダーの名前。Knative のみ、ヘッダーの名前には CloudEvent(**ce-**)プレフィックスが必要です。	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. 依存関係

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

- camel:core
- camel:kamelet

24.3. 用途

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。以下の例では、**ce-foo** ヘッダーを **timer-source** Kamelet からのデータに追加します。ヘッダー名の CloudEvents(**ce-**)プレフィックスが必要です。

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: "ce-foo"
      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、必要に応じて構成を編集します。
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=ce-foo" -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。以下の例では、**timer-source** Kamelet からのデータに **foo** ヘッダーを追加します。

insert-header-action-binding.yaml

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

```
  name: timer-to-kafka
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      period: "10000"
      message: 'msg'
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: insert-header-action
    properties:
      name: "foo"
      value: "The Value"
  sink:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: kafka-sink
    properties:
      bootstrapServers: "my-cluster-kafka-bootstrap.myproject.svc:9092"
      password: "XXX"
      topic: "my-topic"
      user: "XXX"
      securityProtocol: "PLAINTEXT"
```

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てください。

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、必要に応じて構成を編集します。
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=foo" -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/blob/kamelet-catalog-1.6/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. 依存関係

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. 用途

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、必要に応じて構成を編集します。
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、必要に応じて構成を編集します。
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/blob/kamelet-catalog-1.6/is-tombstone-filter-action.kamelet.yaml

第26章 JIRA ソース

重要

Jira Source Kamelet はテクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。

これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、「テクノロジプレビュー機能のサポート範囲」を参照してください。

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	Name (名前)	詳細	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 *	パスワード	Jira にアクセスするためのパスワード	string		
username *	ユーザ名	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. 依存関係

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

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

26.3. 用途

ここでは、**jira-source** の使用方法について説明します。

26.3.1. Knative Source

jira-source KameletをKnativeオブジェクトにバインドすることで、Knativeの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、必要に応じて構成を編集します。
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、必要に応じて構成を編集します。
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/blob/kamelet-catalog-1.6/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	Name (名前)	詳細	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. 依存関係

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. 用途

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、必要に応じて構成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、必要に応じて構成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/blob/kamelet-catalog-1.6/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	Name (名前)	詳細	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. 依存関係

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. 用途

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 Source

jms-amqp-10-source KameletをKnativeオブジェクトにバインドすることで、Knativeの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/blob/kamelet-catalog-1.6/jms-amqp-10-source.kamelet.yaml

第29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

29.1. 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。

29.2. 依存関係

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

29.3. 用途

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

29.3.1. 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
```

29.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **json-deserialize-action-binding.yaml**ファイルをローカルドライブに保存し、必要に応じて構成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
```

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、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。

29.3.2. 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
```

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. **json-deserialize-action-binding.yaml**ファイルをローカルドライブに保存し、必要に応じて構成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
```

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、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。

29.4. KAMELET ソースファイル

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

第30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

30.1. 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。

30.2. 依存関係

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

30.3. 用途

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

30.3.1. 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
```

30.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **json-serialize-action-binding.yaml**ファイルをローカルドライブに保存し、必要に応じて構成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
```

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、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。

30.3.2. 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
```

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. **json-serialize-action-binding.yaml**ファイルをローカルドライブに保存し、必要に応じて構成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
```

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、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。

30.4. KAMELET ソースファイル

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

第31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。

31.1. 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	Name (名前)	詳細	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 *	パスワード	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 *	ユーザ名	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。

31.2. 依存関係

「kafka-sink Kamelet」は、以下の依存関係の有無に依存します。

- camel:kafka
- camel:kamelet

31.3. 用途

ここでは、**kafka-sink**の使用方法について説明します。

31.3.1. 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"
```

31.3.1.1. 前提条件

接続先のOpenShiftクラスタに**「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. **kafka-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成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
```

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。

31.3.2. 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"
```

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. **kafka-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成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
```

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。

31.4. KAMELET ソースファイル

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

第32章 KAFKA SOURCE

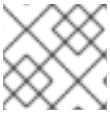
Kafka トピックからデータを受信します。

32.1. 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	Name (名前)	詳細	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 *	ユーザ名	Kafka に対して認証を行うためのユーザー名	string		
Password *	パスワード	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た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。	ブール値	**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。	ブール値	**false**	

プロパティー	Name (名前)	詳細	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 が例外をスロー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	ヘッダーの自動デシリアライズ	Kamelet ソースを有効にすると、すべてのメッセージヘッダーが String 表現にデシリアライズされます。デフォルトは **false** です。	ブール値	**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。

32.2. 依存関係

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

32.3. 用途

ここでは、**kafka-source** の使用方法について説明します。

32.3.1. Knative Source

kafka-source KameletをKnativeオブジェクトにバインドすることで、Knativeの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
```

32.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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。

32.3.2. 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
```

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. **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
```

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。

32.4. KAMELET ソースファイル

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

第33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

33.1. 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	Name (名前)	詳細	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。

33.2. 依存関係

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

- camel:core
- camel:kamelet

33.3. 用途

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。

33.3.1. Knative Action

topic-name-matches-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。

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: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

33.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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、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" 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。

33.3.2. 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
```

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. **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
```

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、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。

33.4. KAMELET ソースファイル

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

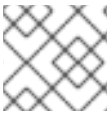
第34章 LOG SINK

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。

34.1. 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	Name (名前)	詳細	型	デフォルト	例
showHeaders	show Headers	受信したヘッダーを表示します。	ブール値	**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）。	ブール値	**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。

34.2. 依存関係

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

- camel:kamelet
- camel:log

34.3. 用途

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。

34.3.1. 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
```

34.3.1.1. 前提条件

接続先のOpenShiftクラスタに**「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. **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
```

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。

34.3.2. 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
```

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. **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
```

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。

34.4. KAMELET ソースファイル

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

第35章 MARIADB SINK

MariaDB データベースにデータを送信する。

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)'

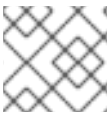
Kamelet は、以下のような入力として受信する必要があります。

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

35.1. 設定オプション

以下の表は、**mariadb-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	Name (名前)	詳細	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 *	パスワード	セキュアな MariaDB 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	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 *	ユーザ名	セキュアな MariaDB 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。

35.2. 依存関係

ランタイム時に、**mariadb-sink** Kamelet は以下の依存関係が存在する必要があります。

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

35.3. 用途

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

35.3.1. 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"
```

35.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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。

35.3.2. Kafka Sink

mariadb-sink Kamelet を Kafka トピックにバインドし、これを 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"
```

35.3.2.1. 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。

35.3.2.2. 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
```

35.3.2.3. 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。

35.4. KAMELET ソースファイル

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

第36章 フィールド動作のマスク

転送中のメッセージの定数値のあるフィールドをマスクする

36.1. 設定オプション

以下の表は、**mask-field-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	Name (名前)	詳細	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	マスクされるフィールドの代替	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。

36.2. 依存関係

ランタイム時に、**mask-field-action** Kamelet は以下の依存関係の有無に依存します。

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

36.3. 用途

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

36.3.1. Knative Action

mask-field-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。

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
```

36.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **mask-field-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 mask-field-action-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、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。

36.3.2. Kafka Action

mask-field-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。

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
```

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. **mask-field-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 mask-field-action-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、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。

36.4. KAMELET ソースファイル

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

第37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。

37.1. 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	Name (名前)	詳細	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。

37.2. 依存関係

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

37.3. 用途

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。

37.3.1. 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
```

37.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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、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。

37.3.2. 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
```

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. **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
```

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、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。

37.4. KAMELET ソースファイル

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/message-timestamp-router-action.kamelet.yaml

第38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。

38.1. 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	Name (名前)	詳細	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	初期設定時にコレクションを作成します（存在しない場合）。	ブール値	**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	Name (名前)	詳細	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。

38.2. 依存関係

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

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

38.3. 用途

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

38.3.1. 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"
```

38.3.1.1. 前提条件

接続先のOpenShiftクラスタに**「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. **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
```

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。

38.3.2. 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"
```

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. **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
```

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。

38.4. KAMELET ソースファイル

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

第39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）。

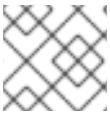
persistentTailTracking オプションが有効になっていない場合、コンシューマーはコレクション全体を消費し、新しいドキュメントが消費するアイドル状態になるのを待ちます。

39.1. 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	Name (名前)	詳細	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	Name (名前)	詳細	型	デフォルト	例
persistentTailTracking	MongoDB Persistent Tail Tracking	永続的なテールトラッキングを有効にします。これは、システムの再起動で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停止した時点からカーソルを復旧します。	ブール値	**false**	
tailTrackIncreasingField	MongoDB Tail Track Increasing フィールド	増加する性質があり、生成されるたびにテールカーソルを配置するために使用される、着信レコードの correlation フィールド。	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。

39.2. 依存関係

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

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

39.3. 用途

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

39.3.1. Knative Source

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
```

39.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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。

39.3.2. 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
```

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. **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
```

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。

39.4. KAMELET ソースファイル

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

第40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)'

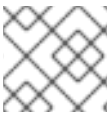
Kamelet は、以下のような入力として受信する必要があります。

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

40.1. 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	Name (名前)	詳細	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 *	パスワード	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 *	ユーザ名	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。

40.2. 依存関係

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
- mvn:mysql:mysql-connector-java

40.3. 用途

ここでは、**mysql-sink**の使用方法について説明します。

40.3.1. 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"
```

40.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **mysql-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成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
```

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 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。

40.3.2. 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"
```

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. **mysql-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成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
```

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 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。

40.4. KAMELET ソースファイル

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

第41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"}'

41.1. 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	Name (名前)	詳細	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 *	パスワード	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 *	ユーザ名	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。

41.2. 依存関係

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

41.3. 用途

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

41.3.1. 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"
```

41.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **postgresql-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成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
```

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。

41.3.2. 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"
```

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. **postgresql-sink-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成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
```

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。

41.4. KAMELET ソースファイル

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

第42章 述語フィルターの動作

JsonPath 式に基づくフィルター

42.1. 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	Name (名前)	詳細	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。

42.2. 依存関係

実行時に **predicate-filter-action** Kamelet は、以下の依存関係の有無に依存します。

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

42.3. 用途

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

42.3.1. 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
```

42.3.1.1. 前提条件

接続先のOpenShiftクラスタに**「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. **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
```

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、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。

42.3.2. 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
```

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. **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
```

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、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。

42.4. KAMELET ソースファイル

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

第43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

43.1. 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	Name (名前)	詳細	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。

43.2. 依存関係

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。

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

43.3. 用途

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

43.3.1. 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":"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: 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; }"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

43.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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、アクションを設定および実行します。

```
kamel bind "timer-source?message={\"first\":\"Ada\",\"last\":\"Lovelace\"}" --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; }" --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。

43.3.2. 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":"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: 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; }"
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: json-serialize-action
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

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. **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
```

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、アクションを設定および実行します。

```
kamel bind "timer-source?message={\"first\":\"Ada\",\"last\":\"Lovelace\"}" --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; }" --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。

43.4. KAMELET ソースファイル

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

第44章 PROTOBUF SERIALIZE アクション

ペイロードを Protobuf にシリアライズする

44.1. 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	Name (名前)	詳細	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。

44.2. 依存関係

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。

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

44.3. 用途

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

44.3.1. 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":"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: 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
```

44.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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={\"first\":\"Ada\",\"last\":\"Lovelace\"}" --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。

44.3.2. 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":"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: 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
```

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. **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
```

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={\"first\":\"Ada\",\"last\":\"Lovelace\"}" --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。

44.4. KAMELET ソースファイル

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

第45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。

45.1. 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	Name (名前)	詳細	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。

45.2. 依存関係

実行時に **regex-router-action** Kamelet は、以下の依存関係の有無に依存します。

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

45.3. 用途

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

45.3.1. 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
```

45.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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 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。

45.3.2. 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
```

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. **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
```

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 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。

45.4. KAMELET ソースファイル

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/regex-router-action.kamelet.yaml

第46章 フィールド置き換えアクション

転送中のメッセージの別のキーに、field を置き換えます。

46.1. 設定オプション

以下の表は、**replace-field-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	Name (名前)	詳細	型	デフォルト	例
disabled *	Disabled	無効にするフィールドのコンマ区切りリスト	string		
enabled *	Enabled	有効にするフィールドのコンマ区切りリスト	string		
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"**

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. 依存関係

ランタイム時に、**replace-field-action** Kamelet は以下の依存関係が存在する必要があります。

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

46.3. 用途

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

46.3.1. Knative Action

replace-field-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。

replace-field-action-binding.yaml

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

```
kind: KameletBinding
metadata:
  name: replace-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: replace-field-action
    properties:
      disabled: "The Disabled"
      enabled: "The Enabled"
      renames: "foo:bar,c1:c2"
  sink:
    ref:
      kind: Channel
      apiVersion: messaging.knative.dev/v1
      name: mychannel
```

46.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **replace-field-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 replace-field-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 timer-source?message=Hello --step replace-field-action -p "step-0.disabled=The
Disabled" -p "step-0.enabled=The Enabled" -p "step-0.renames=foo:bar,c1:c2" channel:mychannel
```

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

replace-field-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。

replace-field-action-binding.yaml

```
apiVersion: camel.apache.org/v1alpha1
kind: KameletBinding
metadata:
  name: replace-field-action-binding
spec:
  source:
    ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: timer-source
    properties:
      message: "Hello"
  steps:
  - ref:
      kind: Kamelet
      apiVersion: camel.apache.org/v1alpha1
      name: replace-field-action
    properties:
      disabled: "The Disabled"
      enabled: "The Enabled"
      renames: "foo:bar,c1:c2"
  sink:
    ref:
      kind: KafkaTopic
      apiVersion: kafka.strimzi.io/v1beta1
      name: my-topic
```

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. **replace-field-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 replace-field-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 timer-source?message=Hello --step replace-field-action -p "step-0.disabled=The
Disabled" -p "step-0.enabled=The Enabled" -p "step-0.renames=foo:bar,c1:c2"
kafka.strimzi.io/v1beta1:KafkaTopic:my-topic
```

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/blob/kamelet-catalog-1.6/replace-field-action.kamelet.yaml

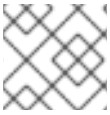
第47章 SALESFORCE ソース

Salesforce から更新を受け取ります。

47.1. 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	Name (名前)	詳細	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 *	パスワード	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 *	ユーザ名	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。

47.2. 依存関係

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

47.3. 用途

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

47.3.1. Knative Source

salesforce-source KameletをKnativeオブジェクトにバインドすることで、Knativeの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
```

47.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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。

47.3.2. 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
```

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. **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
```

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。

47.4. KAMELET ソースファイル

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

第48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。

48.1. 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	Name (名前)	詳細	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	起動ディレクトリー	string		
Password *	パスワード	FTPサーバーにアクセスするためのパスワード	string		
username *	ユーザ名	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	ブール値	**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。

48.2. 依存関係

ランタイム時に、**sftp-sink** Kamelet は、以下の依存関係の有無に依存します。

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

48.3. 用途

本セクションでは、**sftp-sink** を使用する方法を説明します。

48.3.1. 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"
```

48.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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。

48.3.2. 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"
```

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. **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
```

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。

48.4. KAMELET ソースファイル

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

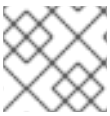
第49章 SFTP ソース

SFTP サーバーからデータを受け取ります。

49.1. 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	Name (名前)	詳細	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	起動ディレクトリー	string		
Password *	パスワード	SFTP サーバーにアクセスするためのパスワード	string		
username *	ユーザ名	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。	ブール値	**true**	
passiveMode	パッシブモード	パッシブモード接続の設定	ブール値	**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。	ブール値	**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。

49.2. 依存関係

ランタイム時に、**sftp-source** Kamelet は、以下の依存関係の有無に依存します。

- camel:ftp

- camel:core
- camel:kamelet

49.3. 用途

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

49.3.1. Knative Source

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
```

49.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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 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。

49.3.2. 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
```

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. **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
```

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 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。

49.4. KAMELET ソースファイル

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

第50章 SLACK ソース

Slack チャネルからメッセージを受信します。

50.1. 設定オプション

以下の表は、**slack-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	Name (名前)	詳細	型	デフォルト	例
チャンネル *	Channel	メッセージの受信元となる Slack チャネル	string		**"#myroom"**
トークン *	トークン	Slack にアクセスするためのトークン。Slack アプリケーションが必要です。このアプリケーションには、channels:history と channels:read のパーミッションが必要です。Bot User OAuth アクセストークンは必要なトークンの種類です。	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。

50.2. 依存関係

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

50.3. 用途

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

50.3.1. Knative Source

slack-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。

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
```

50.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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。

50.3.2. Kafka Source

slack-source Kamelet を Kafka ソースとして使用するには、Kafka トピックにバインドします。

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
```

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. **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
```

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。

50.4. KAMELET ソースファイル

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

第51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"}'

51.1. 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	Name (名前)	詳細	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 *	パスワード	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 *	ユーザ名	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。

51.2. 依存関係

ランタイム時に、**sqlserver-sink** Kamelet は、以下の依存関係の有無に依存します。

- camel:jackson
- camel:kamelet
- camel:sql
- mvn:org.apache.commons:commons-dbcp2:2.7.0
- mvn:com.microsoft.sqlserver:mssql-jdbc:9.2.1.jre11

51.3. 用途

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

51.3.1. 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"
```

51.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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 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。

51.3.2. 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"
```

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. **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
```

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。

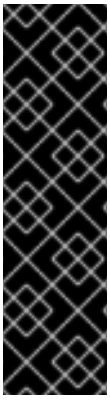
```
kamel bind kafka.strimzi.io/v1beta1:KafkaTopic:my-topic sqlserver-sink -p "sink.databaseName=The Database Name" -p "sink.password=The Password" -p "sink.query=INSERT INTO accounts (username,city) VALUES (:#username,:#city)" -p "sink.serverName=localhost" -p "sink.username=The Username"
```

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。

51.4. KAMELET ソースファイル

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

第52章 TELEGRAM ソース



重要

Telegram Source Kamelet はテクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。

これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、「テクノロジプレビュー機能のサポート範囲」を参照してください。

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

52.1. 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	Name (名前)	詳細	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。

52.2. 依存関係

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

52.3. 用途

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

52.3.1. Knative Source

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
```

52.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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。

52.3.2. 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
```

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. **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
```

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。

52.4. KAMELET ソースファイル

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

第53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。

53.1. 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	Name (名前)	詳細	型	デフォルト	例
message *	Message	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）	整数	**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。

53.2. 依存関係

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

53.3. 用途

ここでは、**timer-source**の使用方法について説明します。

53.3.1. Knative Source

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
```

53.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **timer-source-binding.yaml**ファイルをローカルドライブに保存し、必要に応じて構成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
```

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。

53.3.2. 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
```

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. **timer-source-binding.yaml**ファイルをローカルドライブに保存し、必要に応じて構成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
```

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。

53.4. KAMELET ソースファイル

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

第54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。

54.1. 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	Name (名前)	詳細	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。

54.2. 依存関係

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。

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

54.3. 用途

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

54.3.1. 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
```

54.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **timestamp-router-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成を編集します。
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
```

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、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。

54.3.2. 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
```

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. **timestamp-router-action-binding.yaml** ファイルをローカルドライブに保存し、必要に応じて構成を編集します。
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
```

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、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。

54.4. KAMELET ソースファイル

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/timestamp-router-action.kamelet.yaml

第55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。

55.1. 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	Name (名前)	詳細	型	デフォルト	例
fields *	フィールド	新しいキーを構成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。

55.2. 依存関係

ランタイム時に、**value-to-key-action** Kamelet は、以下の依存関係の有無に依存します。

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

55.3. 用途

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。

55.3.1. 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
```

55.3.1.1. 前提条件

接続先のOpenShiftクラスタに「**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. **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
```

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、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。

55.3.2. 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
```

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. **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
```

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、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。

55.4. KAMELET ソースファイル

https://github.com/openshift-integration/kamelet-catalog/blob/kamelet-catalog-1.6/value-to-key-action.kamelet.yaml