Red Hat Integration 2022.Q1

Quarkus の Camel エクステンションリファレンス

RedHat が提供する Quarkus の Camel エクステンション

Last Updated: 2023-03-25

Red Hat Integration 2022.Q1 Quarkus の Camel エクステンションリファレンス

RedHat が提供する Quarkus の Camel エクステンション

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

Quarkus の Camel エクステンションは、多くの Camel コンポーネントに Quarkus エクステンションを提供します。このリファレンスでは、RedHat でサポートされている各エクステンションの設定について説明します。

目次

前書き

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 エクステンションの概要

1.1. サポートレベルの定義

新たな機能、サービス、およびコンポーネントは、実稼働環境での使用の完全サポートとして Quarkus の Camel エクステンションに含まれるまでに、さまざまなサポートレベルを経ます。これは、オファリングに期待されるエンタープライズグレードの安定性と、お客様やパートナーが新しい Quarkus の Camel エクステンションテクノロジーを試し、今後の開発アクティビティーに役立つフィードバックを提供できるようにすること、の間で適切なバランスを取れるようにするためです。

表1.1 Quarkus の Camel エクステンションのサポートレベル

タイプ	説明
コミュニティーサポート	Red Hat のアップストリームファーストのコミットの一環として、新しいエクステンションの Quarkus の Camel エクステンションディストリビューションへのインテグレーションは、アップストリームコミュニティーから開始します。これらのエクステンションはアップストリームでテストされ、文書化されていますが、弊社はこれらのエクステンションの成熟度を確認しておらず、今後の製品リリースで Red Hat によって正式にサポートされない可能性があります。 **注記** コミュニティーエクステンションは、Camel Quarkus コミュニティープロジェクトのエクステンションリファレンスページにリストされています。
テクノロジープレビュー	テクノロジープレビューの機能は、最新の技術をいち早く提供して、開発段階で機能のテストやフィードバックの収集を可能にするために提供されます。ただし、これらの機能は Red Hat サブスクリプションレベルアグリーメントでは完全にサポートされておらず、機能的に完全でない可能性があり、実稼働環境での使用を目的としていません。Red Hat ではテクノロジープレビュー機能を今後も繰り返し使用することで一般提供に移行できると考えていることから、お客様がこの機能を使用する際に発生する問題の解決に取り組みます。
製品サポート	製品サポートのエクステンションは、Red Hat の公式リリースに含まれており、完全にサポートされます。ドキュメントにギャップはなく、エクステンションはサポートされるすべての構成でテストされます。

1.2. サポートされるエクステンション

33 のエクステンションがあります。

表1.2 Quarkus の Camel エクステンションのサポートマトリックス

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
AWS 2 DynamoDB	camel-quarkus-aws2-ddb	製品サポート	テクノロジープレビュー	AWS SDK バージョン 2.x を使用して、AWS DynamoDB サービスからデータを保存および取得したり、AWS DynamoDB Stream からメッセージを受信したりします。
AWS 2 Kinesis	camel-quarkus-aws2-kinesis	製品サポート	テクノロジープレビュー	AWS SDK バージョン 2.x を使用して、AWS Kinesis Streams からレコードを消費および作成します。
AWS 2 Lambda	camel-quarkus-aws2-lambda	製品サポート	テクノロジープレビュー	AWS SDK バージョン 2.x を使用して、AWS Lambda 関数を管理および呼び出します。
AWS 2 S3 Storage Service	camel-quarkus-aws2-s3	製品サポート	テクノロジープレビュー	AWS SDK バージョン 2.x を使用して、AWS S3 Storage Service からオブジェクトを保存および取得します。
AWS 2 Simple Notification System (SNS)	camel-quarkus-aws2-sns	製品サポート	テクノロジープレビュー	AWS SDK バージョン 2.x を使用して AWS Simple Notification Topic にメッセージを送信します。

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
AWS 2 Simple Queue Service (SQS)	camel-quarkus-aws2-sqs	テクノロジープレビュー	テクノロジープレビュー	AWS SDK バージョン 2.x を使用して AWS SQS サービスを送信先および送信元としてメッセージを送受信します。
Bean	camel-quarkus-bean	製品サポート	テクノロジープレビュー	Java Bean のメソッドを呼び出します。
Core	camel-quarkus-core	製品サポート	テクノロジープレビュー	Camel コア機能と基本的な Camel 言語: Constant、ExchangeProperty、Header、Ref、Ref、Simple および Tokeinze
Direct	camel-quarkus-direct	製品サポート	テクノロジープレビュー	同じ Camel コンテキストから別のエンドポイントを同期的に呼び出します。
Elasticsearch Rest	camel-quarkus-elasticsearch-rest	テクノロジープレビュー	テクノロジープレビュー	REST API 経由で ElasticSearch を使用してリクエストを送信します。
File	camel-quarkus-file	製品サポート	テクノロジープレビュー	ファイル読み取りおよび書き込みます。
FTP	camel-quarkus-ftp	製品サポート	テクノロジープレビュー	FTP または SFTP サーバーとの間で、ファイルをアップロードおよびダウンロードします。
HTTP	camel-quarkus-http	製品サポート	テクノロジープレビュー	Apache HTTP Client 4.x を使用して、外部の HTTP サーバーにリクエストを送信します。

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
Jira	camel-quarkus-jira	テクノロジープレビュー	テクノロジープレビュー	JIRA 問題トラッカーと対話します。
JMS	camel-quarkus-jms	製品サポート	テクノロジープレビュー	JMS Queue または Topic との間でメッセージを送受信します。
JTA	camel-quarkus-jta	製品サポート	テクノロジープレビュー	Java Transaction API (JTA) および Narayana トランザクションマネージャーを使用して、Camel ルートをトランザクションに含めます。
Kafka	camel-quarkus-kafka	製品サポート	テクノロジープレビュー	Apache Kafka ブローカーとの間でメッセージを送受信します。
Kamelet	camel-quarkus-kamelet	製品サポート	テクノロジープレビュー	Kamelet コンポーネントは、Camel Route Template エンジンとの対話をサポートします。
Log	camel-quarkus-log	製品サポート	テクノロジープレビュー	基礎となるロギングメカニズムにメッセージをログとして記録します。
MicroProfile Health	camel-quarkus-microprofile-health	製品サポート	テクノロジープレビュー	Eclipse MicroProfile Health と Camel ヘルスチェックをブリッジングします。
MicroProfile Metrics	camel-quarkus-microprofile-metrics	製品サポート	テクノロジープレビュー	Camel ルートからメトリクスを公開します。

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
MLLP	camel-quarkus-mllp	製品サポート	テクノロジープレビュー	MLLP プロトコルを使用して外部システムと通信します。
Mock	camel-quarkus-mock	製品サポート	テクノロジープレビュー	モックを使用してルートおよび仲介ルールをテストします。
MongoDB	camel-quarkus-mongodb	テクノロジープレビュー	テクノロジープレビュー	MongoDB ドキュメントおよびコレクションの操作を実行します。
Netty	camel-quarkus-netty	製品サポート	テクノロジープレビュー	Netty 4.x で TCP または UDP を使用するソケットレベルのネットワーク。
OpenAPI Java	camel-quarkus-openapi-java	製品サポート	テクノロジープレビュー	OpenAPI ドキュメントを使用するための REST-dsl サポート
Platform HTTP	camel-quarkus-platform-http	製品サポート	テクノロジープレビュー	現在のプラットフォームで利用可能な HTTP サーバーを使用して HTTP エンドポイントを公開します。
Rest	camel-quarkus-rest	製品サポート	テクノロジープレビュー	REST サービスおよび OpenAPI Specification を公開するか、外部の REST サービスを呼び出します。
Salesforce	camel-quarkus-salesforce	製品サポート	テクノロジープレビュー	Java DTO を使用して Salesforce と通信します。

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
SEDA	camel-quarkus-seda	製品サポート	テクノロジープレビュー	同じ JVM の Camel コンテキストから別のエンドポイントを非同期に呼び出します。
SQL	camel-quarkus-sql	製品サポート	テクノロジープレビュー	Spring JDBC を使用して SQL クエリーを実行します。
Timer	camel-quarkus-timer	製品サポート	テクノロジープレビュー	java.util.Timer を使用して指定した間隔でメッセージを生成します。
XQuery	camel-quarkus-saxon	製品サポート	テクノロジープレビュー	XQuery および Saxon を使用して XML ペイロードをクエリーまたは変換します。

1.3. サポートされるデータフォーマット

データフォーマットは 8 つあります。

表1.3 Quarkus の Camel エクステンションのサポートマトリックス

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
Avro	camel-quarkus-avro	製品サポート	テクノロジープレビュー	Apache Avro バイナリーデータフォーマットを使用して、メッセージをシリアライズおよびデシリアライズします。
Avro Jackson	camel-quarkus-jackson-avro	製品サポート	テクノロジープレビュー	Jackson を使用して、POJO を Avro にマーシャリングし、戻します。

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
Bindy	camel-quarkus-bindy	製品サポート	テクノロジープレビュー	Camel Bindy Marshal を使用して POJO とコンマ区切り値 (CSV) フォーマットの間をマーシャリングおよびアンマーシャリングし、Camel Bindy Marshal を使用して POJO と固定フィールド長フォーマットの間をアンマーシャリングし、Camel Bindy を使用して POJO とキー/値ペア (KVP) フォーマットの間をアンマーシャリングします。
HL7	camel-quarkus-hl7	製品サポート	テクノロジープレビュー	HL7 MLLP コーデックを使用して、HL7 (Health Care) モデルオブジェクトをマーシャリングおよびアンマーシャリングします。
Jackson	camel-quarkus-jackson	製品サポート	テクノロジープレビュー	Jackson を使用して、POJO を JSON にマーシャリングし、戻します。
JacksonXML	camel-quarkus-jacksonxml	製品サポート	テクノロジープレビュー	Jackson の XMLMapper エクステンションを使用して、XML ペイロードを POJO にアンマーシャリングし、戻します。
Protobuf Jackson	camel-quarkus-jackson-protobuf	製品サポート	テクノロジープレビュー	Jackson を使用して、POJO を Protobuf にマーシャリングし、戻します。

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
SOAP dataformat	camel-quarkus-soap	製品サポート	テクノロジープレビュー	Java オブジェクトを SOAP メッセージにマーシャリングし、戻します。

1.4. サポートされる言語

12 つの言語があります。

表1.4 Quarkus の Camel エクステンションのサポートマトリックス

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
Bean メソッド	camel-quarkus-bean	製品サポート	テクノロジープレビュー	Java Bean のメソッドを呼び出します。
Constant	camel-quarkus-core	製品サポート	テクノロジープレビュー	固定の値は、ルートの起動時に一度だけ設定されます。
ExchangeProperty	camel-quarkus-core	製品サポート	テクノロジープレビュー	指定された Camel Exchange プロパティーの値を取得します。
File	camel-quarkus-core	製品サポート	テクノロジープレビュー	File/simple 言語を使用した式および述語の場合。
Header	camel-quarkus-core	製品サポート	テクノロジープレビュー	指定された Camel Message ヘッダーの値を取得します。
HL7 Terser	camel-quarkus-hl7	製品サポート	テクノロジープレビュー	HL7 MLLP コーデックを使用して、HL7 (Health Care) モデルオブジェクトをマーシャリングおよびアンマーシャリングします。

エクステンション	アーティファクト	JVM サポートレベル	ネイティブサポートレベル	説明
Ref	camel-quarkus-core	製品サポート	テクノロジープレビュー	Camel Registry の式を検索して評価します。
Simple	camel-quarkus-core	製品サポート	テクノロジープレビュー	Camel Exchange に対して、Camel の組み込み Simple 言語式を評価します。
Tokenize	camel-quarkus-core	製品サポート	テクノロジープレビュー	指定の区切り文字パターンを使用して、テキストペイロードをトークン化します。
JSON Path	camel-quarkus-jsonpath	製品サポート	テクノロジープレビュー	JSON メッセージのボディーに対して、JsonPath 式を評価します。
XPath	camel-quarkus-xpath	製品サポート	テクノロジープレビュー	XML ペイロードに対して XPath 式を評価します。
XQuery	camel-quarkus-saxon	製品サポート	テクノロジープレビュー	XQuery および Saxon を使用して XML ペイロードをクエリーまたは変換します。

第2章 エクステンションの参照情報

この章では、Quarkus の Camel エクステンションに関するリファレンス情報を提供します。

重要

このテクノロジープレビューリリースには、利用可能な Camel Quarkus エクステンションのターゲットサブセットが含まれています。将来のリリースでは、Quarkus の Camel エクステンションに追加のエクステンションが追加される予定です。

2.1. AVRO

Apache Avro バイナリーデータフォーマットを使用して、メッセージをシリアライズおよびデシリアライズします。

2.1.1. 含まれるもの

- Avro データフォーマット

使用方法と設定の詳細については、上記リンクを参照してください。

2.1.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-avro</artifactId>
</dependency>
```

2.1.3. 追加の Camel Quarkus 設定

vanilla Camel から知られている標準的な使用方法以外に、Camel Quarkus では、JVM と Native モードの両方でビルド時に Avro スキーマを解析する機能が追加されています。

Camel Quarkus 2.0.0 以降、Avro スキーマファイルから Avro クラスを生成するための推奨されるアプローチは、**quarkus-avro** エクステンションによって作成されたものです。以下が必要です。

1. ***.avsc**ファイルを**src/main/avro**または**src/test/avro**という名前のフォルダーに保存します
2. **quarkus-maven-plugin**の通常の**ビルド**目標に加えて、**generate-code**目標を追加します。

```
<plugin>
    <groupId>${quarkus.platform.group-id}</groupId>
    <artifactId>quarkus-maven-plugin</artifactId>
    <executions>
        <execution>
            <id>generate-code-and-build</id>
            <goals>
                <goal>generate-code</goal>
                <goal>build</goal>
```

```
            </goals>
          </execution>
        </executions>
      </plugin>
```

Camel Quarkus Avro 統合テスト およびQuarkus Avro 統合テスト の動作設定を参照してください。

2.1.3.1. 非推奨: @BuildTimeAvroDataFormat アノテーション。

Camel Quarkus 2.0.0 より前は、**@BuildTimeAvroDataFormat** アノテーションが、Avro スキーマファイルから Avro エンティティーを生成する推奨される方法でした。

以下の例では、**user.avsc**スキーマリソースが最初にビルド時に解析されます。次に、以前に解析されたスキーマを使用する **AvroDataFormat** インスタンスが、実行時に **buildTimeAvroDataFormat** フィールドに挿入されます。最終的に、受信メッセージをマーシャリングするために、注入されたデータフォーマットを **configure()** メソッドから使用します。

```
import org.apache.camel.quarkus.component.avro.BuildTimeAvroDataFormat;
...
@BuildTimeAvroDataFormat("user.avsc")
AvroDataFormat buildTimeAvroDataFormat;

@Override
public void configure() {
  from("direct:marshalUsingBuildTimeAvroDataFormat").marshal(buildTimeAvroDataFormat);
}
```

Camel Quarkus 2.0.0 以降、@BuildTimeAvroDataFormat は非推奨になりました。quarkus-avro からのビルド時間クラス生成アプローチが推奨されます。そのため、ビルド時に quarkus-avro によって @AvroGenerated クラスが作成されるように、*.avsc ファイルを avro という名前のフォルダーに保存することが推奨されます。

Camel Quarkus Avro 統合テスト で動作している実行設定を参照してください。ここ には、quarkus-avro 統合テストもあります。

2.2. AWS 2 DYNAMODB

AWS SDK バージョン 2.x を使用して、AWS DynamoDB サービスからデータを保存および取得したり、AWS DynamoDB Stream からメッセージを受信したりします。

2.2.1. 含まれるもの

- AWS DynamoDB コンポーネント、URI 構文: **aws2-ddb:tableName**
- AWS DynamoDB Streams コンポーネント、URI 構文: **aws2-ddbstream:tableName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.2.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-aws2-ddb</artifactId>
</dependency>
```

2.2.3. ネイティブモードの SSL

このエクステンションは、ネイティブモードでの SSL サポートを自動的に有効にします。したがって、自分で **quarkus.ssl.native=true** を **application.properties** に追加する必要はありません。「Quarkus SSL ガイド」も参照してください。

2.2.4. 追加の Camel Quarkus 設定

2.2.4.1. Quarkus Amazon DynamoDB とのオプションの統合

必要に応じて、Quarkus Amazon DynamoDB エクステンションを Camel Quarkus AWS 2 DynamoDB と組み合わせて使用することができます。これは完全に任意であり、必須ではないことに注意してください。Quarkus のドキュメント に従ってください。ただし、次の注意事項に注意してください。

1. クライアントタイプ **apache** は、次のプロパティーを設定して選択する必要があります。

```
quarkus.dynamodb.sync-client.type=apache
```

2. **DynamoDbClient** は、Camel Quarkus が実行時に検索できるように、 Quarkus CDI 参照 の意味で「削除不可」である必要があります。たとえば、**DynamoDbClient**を注入するダミー Bean を追加することで、これに到達できます。

```
import javax.enterprise.context.ApplicationScoped;
import io.quarkus.arc.Unremovable;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

@ApplicationScoped
@Unremovable
class UnremovableDynamoDbClient {
    @Inject
    DynamoDbClient dynamoDbClient;
}
```

2.3. AWS 2 KINESIS

AWS SDK バージョン 2.x を使用して、AWS Kinesis Streams からレコードを消費および作成します。

2.3.1. 含まれるもの

- AWS Kinesis コンポーネント、URI 構文: **aws2-kinesis:streamName**
- AWS Kinesis Firehose コンポーネント、URI 構文: **aws2-kinesis-firehose:streamName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.3.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-aws2-kinesis</artifactId>
</dependency>
```

2.3.3. ネイティブモードの SSL

このエクステンションは、ネイティブモードでの SSL サポートを自動的に有効にします。したがって、自分で **quarkus.ssl.native=true** を **application.properties** に追加する必要はありません。
「Quarkus SSL ガイド」も参照してください。

2.4. AWS 2 LAMBDA

AWS SDK バージョン 2.x を使用して、AWS Lambda 関数を管理および呼び出します。

2.4.1. 含まれるもの

- AWS Lambda コンポーネント、URI 購買: **aws2-lambda:function**

使用方法と設定の詳細については、上記リンクを参照してください。

2.4.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-aws2-lambda</artifactId>
</dependency>
```

2.4.3. Camel Quarkus の制限

getAlias および **listAliases** 操作を機能させるには、**pojoRequest** で使用する必要があります。これは、これらの操作の要求は、以下に示すように手動で明示的に作成する必要があることを意味します。

getAlias 要求を手動で作成する例:

```
.process(new Processor() {
   public void process(Exchange exchange) {
    GetAliasRequest getAliasRequest =
GetAliasRequest.builder().functionName(functionName).name(aliasName).build();
    exchange.getIn().setBody(getAliasRequest);
  }})
.to("aws2-lambda:functionName?operation=getAlias&pojoRequest=true");
```

手動で**listAliases**要求を作成する例:

```
.process(new Processor() {
   public void process(Exchange exchange) {
    ListAliasesRequest listAliasesRequest =
ListAliasesRequest.builder().functionName(functionName).build();
    exchange.getIn().setBody(listAliasesRequest);
  }})
.to("aws2-lambda:functionName?operation=listAliases&pojoRequest=true");
```

2.4.4. ネイティブモードの SSL

このエクステンションは、ネイティブモードでの SSL サポートを自動的に有効にします。したがって、自分で **quarkus.ssl.native=true** を **application.properties** に追加する必要はありません。「Quarkus SSL ガイド」も参照してください。

2.4.5. 追加の Camel Quarkus 設定

2.4.5.1. Camel aws2-lambda エクステンションによって quarkus-amazon-lambda を活用することはできません

quarkus-amazon-lambdaエクステンションを使用すると、Quarkus を使用して AWS Lambda をビルドできますが、Camel コンポーネントは既存の関数を管理 (デプロイ、アンデプロイなど) します。したがって、**quarkus-amazon-lambda** を **aws2-lambda** エクステンションのクライアントとして使用することはできません。

2.5. AWS 2 S3 STORAGE SERVICE

AWS SDK バージョン 2.x を使用して、AWS S3 Storage Service からオブジェクトを保存および取得します。

2.5.1. 含まれるもの

- AWS S3 Storage Service コンポーネント、URI 構文: **aws2-s3://bucketNameOrArn**

使用方法と設定の詳細については、上記リンクを参照してください。

2.5.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
   <groupId>org.apache.camel.quarkus</groupId>
   <artifactId>camel-quarkus-aws2-s3</artifactId>
</dependency>
```

2.5.3. ネイティブモードの SSL

このエクステンションは、ネイティブモードでの SSL サポートを自動的に有効にします。したがって、自分で **quarkus.ssl.native=true** を **application.properties** に追加する必要はありません。「Quarkus SSL ガイド」も参照してください。

2.5.4. 追加の Camel Quarkus 設定

2.5.4.1. Quarkus Amazon S3 とのオプションの統合

必要に応じて、Quarkus Amazon S3 エクステンションを Camel Quarkus AWS 2 S3 Storage Service と組み合わせて使用することができます。これは完全に任意であり、必須ではないことに注意してください。Quarkus ドキュメント に従ってください。ただし、次の注意事項に注意してください。

1. クライアントタイプ **apache** は、次のプロパティーを設定して選択する必要があります。

```
quarkus.s3.sync-client.type=apache
```

2. **S3Client** は、Camel Quarkus が実行時に検索できるように、 Quarkus CDI 参照 の意味で「削除不可」である必要があります。たとえば、**S3Client** を注入するダミー Bean を追加することで、これに到達できます。

```
import javax.enterprise.context.ApplicationScoped;
import io.quarkus.arc.Unremovable;
import software.amazon.awssdk.services.s3.S3Client;

@ApplicationScoped
@Unremovable
class UnremovableS3Client {
    @Inject
    S3Client s3Client;
}
```

2.6. AWS 2 SIMPLE NOTIFICATION SYSTEM (SNS)

AWS SDK バージョン 2.x を使用して AWS Simple Notification Topic にメッセージを送信します。

2.6.1. 含まれるもの

- AWS Simple Notification System (SNS) コンポーネント 、URI 構文: **aws2-sns:topicNameOrArn**

使用方法と設定の詳細については、上記リンクを参照してください。

2.6.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-aws2-sns</artifactId>
</dependency>
```

2.6.3. ネイティブモードの SSL

このエクステンションは、ネイティブモードでの SSL サポートを自動的に有効にします。したがって、自分で **quarkus.ssl.native=true** を **application.properties** に追加する必要はありません。「Quarkus SSL ガイド」も参照してください。

2.6.4. 追加の Camel Quarkus 設定

2.6.4.1. Quarkus Amazon SNS とのオプションの統合

必要に応じて、Quarkus Amazon SNS エクステンションを Camel Quarkus AWS 2 Simple Notification System (SNS) と組み合わせて使用することができます。これは完全に任意であり、必須ではないことに注意してください。Quarkus ドキュメント に従ってください。ただし、次の注意事項に注意してください。

1. クライアントタイプ **apache** は、次のプロパティーを設定して選択する必要があります。

```
quarkus.sns.sync-client.type=apache
```

2. **SnsClient** は、Camel Quarkus が実行時に検索できるように、 Quarkus CDI 参照 の意味で「削除不可」にする必要があります。たとえば、**SnsClient** を注入するダミー Bean を追加することで、これに到達できます。

```
import javax.enterprise.context.ApplicationScoped;
import io.quarkus.arc.Unremovable;
import software.amazon.awssdk.services.sns.SnsClient;

@ApplicationScoped
@Unremovable
class UnremovableSnsClient {
    @Inject
    SnsClient snsClient;
}
```

2.7. AWS 2 SIMPLE QUEUE SERVICE (SQS)

AWS SDK バージョン 2.x を使用して AWS SQS サービスを送信先および送信元としてメッセージを送受信します。

2.7.1. 含まれるもの

- AWS Simple Queue Service (SQS) コンポーネント、URI 構文: **aws2-sqs:queueNameOrArn**

使用方法と設定の詳細については、上記リンクを参照してください。

2.7.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-aws2-sqs</artifactId>
</dependency>
```

2.7.3. ネイティブモードの SSL

このエクステンションは、ネイティブモードでの SSL サポートを自動的に有効にします。したがって、自分で **quarkus.ssl.native=true** を **application.properties** に追加する必要はありません。「Quarkus SSL ガイド」も参照してください。

2.7.4. 追加の Camel Quarkus 設定

2.7.4.1. Quarkus Amazon SQS とのオプションの統合

必要に応じて、Quarkus Amazon SQS エクステンションを Camel Quarkus AWS 2 Simple Queue Service (SQS) と組み合わせて使用することができます。これは完全に任意であり、必須ではないことに注意してください。Quarkus ドキュメント に従ってください。ただし、次の注意事項に注意してください。

1. クライアントタイプ **apache** は、次のプロパティーを設定して選択する必要があります。

```
quarkus.sqs.sync-client.type=apache
```

2. **SqsClient** は、Camel Quarkus が実行時に検索できるように、 Quarkus CDI 参照 の意味で「削除不可」にする必要があります。たとえば、**SqsClient** を注入するダミー Bean を追加することで、これに到達できます。

```
import javax.enterprise.context.ApplicationScoped;
import io.quarkus.arc.Unremovable;
import software.amazon.awssdk.services.sqs.SqsClient;

@ApplicationScoped
@Unremovable
class UnremovableSqsClient {
    @Inject
    SqsClient sqsClient;
}
```

2.8. BEAN

Java Bean のメソッドを呼び出します。

2.8.1. 含まれるもの

- Bean コンポーネント、URI 構文: **bean:beanName**
- Bean メソッド言語
- Class コンポーネント、URI 構文: **class:beanName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.8.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

-

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-bean</artifactId>
</dependency>
```

2.8.3. 用途

Camel レジストリーで使用できる Bean のメソッドを呼び出す場合を除き、Bean コンポーネントおよび Bean メソッド言語は、Quarkus CDI Bean を呼び出すこともできます。

2.9. BINDY

Camel Bindy を使用して、片側の POJO と、反対側のコンマ区切り値 (CSV)、固定フィールド長、またはキーと値のペア (KVP) 形式の間のマーシャリングとアンマーシャル

2.9.1. 含まれるもの

- Bindy CSV データフォーマット
- Bindy 固定長データフォーマット
- Bindy キー/値ペアデータフォーマット

使用方法と設定の詳細については、上記リンクを参照してください。

2.9.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-bindy</artifactId>
</dependency>
```

2.9.3. Camel Quarkus の制限

ネイティブモードで camel-quarkus-bindy を使用する場合は、ビルドマシンのロケールのみがサポートされます。

たとえば、ロケールが french のビルドマシンの場合、以下のコード

```
BindyDataFormat dataFormat = new BindyDataFormat();
dataFormat.setLocale("ar");
```

は、JVM モードでは期待どおりに数値をアラビア数字にフォーマットします。ただし、ネイティブモードでは数値をフランス語的にフォーマットします。

さらに調整しないと、ビルドマシンのデフォルトロケールが使用されます。別のロケールは、quarkus.native.user-language および quarkus.native.user-country 設定プロパティーで指定できます。

2.10. コア

Camel コア機能と基本的な Camel 言語/ Constant、ExchangeProperty、Header、Ref、Ref、Simple および Tokeinze

2.10.1. 含まれるもの

- Constant 言語
- ExchangeProperty 言語
- File 言語
- Header 言語
- Ref 言語
- Simple 言語
- Tokenize 言語

使用方法と設定の詳細については、上記リンクを参照してください。

2.10.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-core</artifactId>
</dependency>
```

2.10.3. 追加の Camel Quarkus 設定

2.10.3.1. Simple 言語

2.10.3.1.1. OGNL 表記の使用

Simple 言語から OGNL 表記を使用する場合は、**camel-quarkus-bean** エクステンションを使用する必要があります。

たとえば、以下の簡易式は、**Client** 型のメッセージボディーの **getAddress()** メソッドにアクセスします。

```
---
simple("${body.address}")
---
```

このような場合、ここで説明される ように、camel-quarkus-bean エクステンションで追加の依存関係を取る必要があります。ネイティブモードでは、反映するためにいくつかのクラスを登録する必要がある場合があります。上記の例では、**Client** クラスを 反映するために登録する 必要があります。

2.10.3.1.2. ネイティブモードでの動的型解決の使用

${mandatoryBodyAs(TYPE)}**、**${type:package.Enum.CONSTANT}、または **${body} is TYPE** などの単純な式から型を動的に解決する場合は、反映するために手動で一部のクラスを登録する必要がある場合があります。

たとえば、以下の単純な式は、ランタイム時に **java.nio.ByteBuffer** 型を動的に解決します。

```
---
simple("${body} is 'java.nio.ByteBuffer'")
---
```

したがって、**java.nio.ByteBuffer** クラスを 反映するために登録する 必要があります。

2.10.3.1.3. ネイティブモードでのクラスパスリソースと Simple 言語の使用

次の例のように、ルートがクラスパスから Simple スクリプトをロードすることになっている場合

```
from("direct:start").transform().simple("resource:classpath:mysimple.txt");
```

次に、Quarkus **quarkus.native.resources.includes** プロパティーを使用して、以下に示すようにネイティブ実行可能ファイルにリソースを含める必要があります。

```
quarkus.native.resources.includes = mysimple.txt
```

2.10.3.1.4. ネイティブモードのプロパティーを介したカスタム Bean の設定

#class:* や **#type:*** などの設定でネイティブモードのプロパティーを介してカスタム Bean を指定する場合は、リフレクション用にいくつかのクラスを手動で登録する必要がある場合があります。

たとえば、以下のカスタム Bean 定義には、Bean のインスタンス化とセッターの呼び出しにリフレクションを使用することが含まれます。

```
---
camel.beans.customBeanWithSetterInjection =
#class:org.example.PropertiesCustomBeanWithSetterInjection
camel.beans.customBeanWithSetterInjection.counter = 123
---
```

そのため、クラス **PropertiesCustomBeanWithSetterInjection** を リフレクション用に登録 する必要があります。この場合は、フィールドアクセスを省略できることに注意してください。

設定プロパティー	タイプ	デフォルト
quarkus.camel.bootstrap.enabled true に設定すると、**CamelRuntime** は自動的に起動します。	**boolean**	**true**

設定プロパティー	タイプ	デフォルト
quarkus.camel.service.discovery.exclude-patterns クラスパスの Camel サービス定義ファイルと一致する Ant-path フォーマットのパターンのコンマ区切りリスト。一致するファイルで定義されたサービスは、**org.apache.camel.spi.FactoryFinder** メカニズムで検出されません。除外は包含よりも優先されます。ここで定義された除外は、Camel Quarkus エクステンションで追加されたサービスの検出性を拒否するのにも使用できます。値の例: **META-INF/services/org/apache/camel/foo/*,META-INF/services/org/apache/camel/foo/**/bar**	**string**	
quarkus.camel.service.discovery.include-patterns クラスパスの Camel サービス定義ファイルと一致する Ant-path フォーマットのパターンのコンマ区切りリスト。指定したファイルが **exclude-patterns** で除外されない限り、一致するファイルで定義されたサービスは、**org.apache.camel.spi.FactoryFinder** メカニズムで検出されます。Camel Quarkus エクステンションには、デフォルトで一部のサービスが含まれる可能性があることに注意してください。ここで選択されそれらのサービスに追加されたサービスおよび **exclude-patterns** で定義された除外は、ユニオンセットに適用されます。値の例: **META-INF/services/org/apache/camel/foo/*,META-INF/services/org/apache/camel/foo/**/bar**	**string**	
quarkus.camel.service.registry.exclude-patterns クラスパスの Camel サービス定義ファイルと一致する Ant-path フォーマットのパターンのコンマ区切りリスト。一致するファイルで定義されたサービスは、アプリケーションの静的初期化中に Camel レジストリーに追加されません。除外は包含よりも優先されます。ここで定義された除外は、Camel Quarkus エクステンションで追加されたサービスの登録を拒否するのにも使用できます。値の例: **META-INF/services/org/apache/camel/foo/*,META-INF/services/org/apache/camel/foo/**/bar**	**string**	

設定プロパティー	タイプ	デフォルト
quarkus.camel.service.registry.include-patterns クラスパスの Camel サービス定義ファイルと一致する Ant-path フォーマットのパターンのコンマ区切りリスト。指定したファイルが **exclude-patterns** で除外されない限り、一致するファイルで定義されたサービスは、アプリケーションの静的初期化中に Camel レジストリーに追加されます。Camel Quarkus エクステンションには、デフォルトで一部のサービスが含まれる可能性があることに注意してください。ここで選択されそれらのサービスに追加されたサービスおよび **exclude-patterns** で定義された除外は、ユニオンセットに適用されます。値の例: **META-INF/services/org/apache/camel/foo/*,META-INF/services/org/apache/camel/foo/**/bar**	**string**	
quarkus.camel.runtime-catalog.components **true** の場合、アプリケーションに埋め込まれた Runtime Camel Catalog には、アプリケーションで利用可能な Camel コンポーネントの JSON スキーマが含まれます。それ以外の場合は、コンポーネントの JSON スキーマは Runtime Camel Catalog で利用可能ではなく、アクセスしようとすると RuntimeException が発生します。これを **false** に設定すると、ネイティブイメージのサイズを縮小することができます。JVM モードでは、ネイティブモードとの動作の一貫性を確立する場合を除き、このフラグを **false** に設定することは実際の利点がありません。	**boolean**	**true**
quarkus.camel.runtime-catalog.languages **true** の場合、アプリケーションに埋め込まれた Runtime Camel Catalog には、アプリケーションで利用可能な Camel 言語の JSON スキーマが含まれます。それ以外の場合は、言語の JSON スキーマは Runtime Camel Catalog で利用可能ではなく、アクセスしようとすると RuntimeException が発生します。これを **false** に設定すると、ネイティブイメージのサイズを縮小することができます。JVM モードでは、ネイティブモードとの動作の一貫性を確立する場合を除き、このフラグを **false** に設定することは実際の利点がありません。	**boolean**	**true**
quarkus.camel.runtime-catalog.dataformats **true** の場合、アプリケーションに埋め込まれた Runtime Camel Catalog には、アプリケーションで利用可能な Camel データフォーマットの JSON スキーマが含まれます。それ以外の場合は、データフォーマットの JSON スキーマは Runtime Camel Catalog で利用可能ではなく、アクセスしようとすると RuntimeException が発生します。これを **false** に設定すると、ネイティブイメージのサイズを縮小することができます。JVM モードでは、ネイティブモードとの動作の一貫性を確立する場合を除き、このフラグを **false** に設定することは実際の利点がありません。	**boolean**	**true**

設定プロパティー	タイプ	デフォルト
quarkus.camel.runtime-catalog-models **true** の場合、アプリケーションに埋め込まれた Runtime Camel Catalog には、アプリケーションで利用可能な Camel EIP モデルの JSON スキーマが含まれます。それ以外の場合は、EIP モデルの JSON スキーマは Runtime Camel Catalog で利用可能ではなく、アクセスしようとすると RuntimeException が発生します。これを **false** に設定すると、ネイティブイメージのサイズを縮小することができます。JVM モードでは、ネイティブモードとの動作の一貫性を確立する場合を除き、このフラグを **false** に設定することは実際の利点がありません。	**boolean**	**true**
quarkus.camel.routes-discovery.enabled 静的な初期化時のルートの自動検出を有効にします。	**boolean**	**true**
quarkus.camel.routes-discovery.exclude-patterns RouteBuilder クラスの除外フィルタースキャンに使用されます。除外フィルタリングは、包含フィルターよりも優先されます。パターンは Ant-path スタイルのパターンを使用しています。複数のパターンをコンマで区切って指定することができます。たとえば、Bar から始まるすべてのクラスを除外するには、**/Bar* を使用します。特定のパッケージからのすべてのルートを除外するには、com/mycompany/bar/* を使用します。特定のパッケージおよびそのサブパッケージからのすべてのルートを除外するには、2 つのワイルドカードを使用します (com/mycompany/bar/**)。特定の 2 つのパッケージからのすべてのルートを除外するには、com/mycompany/bar/*,com/mycompany/stuff/* を使用します。	**string**	
quarkus.camel.routes-discovery.include-patterns RouteBuilder クラスの包含フィルタースキャンに使用されます。除外フィルタリングは、包含フィルターよりも優先されます。パターンは Ant-path スタイルのパターンを使用しています。複数のパターンをコンマで区切って指定することができます。たとえば、Foo から始まるすべてのクラスを含めるには、**/Foo* を使用します。特定のパッケージからのすべてのルートを含めるには、com/mycompany/foo/* を使用します。特定のパッケージおよびそのサブパッケージからのすべてのルートを含めるには、2 つのワイルドカードを使用します (com/mycompany/foo/**)。特定の 2 つのパッケージからのすべてのルートを含めるには、com/mycompany/foo/*,com/mycompany/stuff/* を使用します。	**string**	

設定プロパティー	タイプ	デフォルト
quarkus.camel.native.resources.exclude-patterns Camel Quarkus 2.0.0 では **quarkus.native.resources.excludes** に置き換えられました。このプロパティーを使用すると、ビルド時に例外が出力されます。	**string**	
quarkus.camel.native.resources.include-patterns Camel Quarkus 2.0.0 では、**quarkus.native.resources.includes** に置き換えられました。このプロパティーを使用すると、ビルド時に例外が出力されます。	**string**	
quarkus.camel.native.reflection.exclude-patterns 反映のために登録から除外されるクラス名に一致する Ant-path フォーマットのパターンのコンマ区切りリスト。**java.lang.Class.getName()** メソッドによって返されるままのクラス名フォーマットを使用します。パッケージセグメントはピリオド **.** で区切られ、内部クラスはドル記号 **$** で区切られます。このオプションは、**include-patterns** により選択されたセットを絞り込みます。デフォルトでは、クラスは除外されません。このオプションは、Quarkus エクステンションによって内部で登録されたクラスの登録解除には使用できません。	**string**	

設定プロパティー	タイプ	デフォルト
quarkus.camel.native.reflection.include-patterns 反映のために登録されるクラス名に一致する Ant-path フォーマットのパターンのコンマ区切りリスト。**java.lang.Class.getName()** メソッドによって返されるままのクラス名フォーマットを使用します。パッケージセグメントはピリオド **.** で区切られ、内部クラスはドル記号 **$** で区切られます。デフォルトでは、クラスは含まれません。このオプションで選択されるセットは、**exclude-patterns** により絞り込むことができます。Quarkus エクステンションは通常、反映のために必要なクラスを登録することに注意してください。このオプションは、組み込みの機能では十分ではない場合に有用です。このオプションは、コンストラクター、フィールド、およびメソッドの完全な反映アクセスを有効にします。よりきめ細かい制御が必要な場合は、Java コードで **io.quarkus.runtime.annotations.RegisterForReflection** アノテーションを使用することを検討してください。このオプションが正しく機能するには、次の条件の少なくとも 1 つが満たされている必要があります。- パターンにワイルドカード (*または/) がない - 選択したクラスを含むアーティファクトに Jandex インデックス (**META-INF/jandex.idx**) が含まれている) - 選択したクラスを含むアーティファクトは、**application.properties** のオプションの **quarkus.index-dependency.*** ファミリーを使用してインデックスに登録されます - 例: quarkus.index-dependency.my-dep.group-id = org.my-group quarkus.index-dependency.my-dep.artifact-id = my-artifact ここで、**my-dep** は、**org.my-group** と **my-artifact** が一緒に属していることを Quarkus に伝えるために選択したラベルです。	**string**	
quarkus.camel.native.reflection.serialization-enabled **true** の場合、基本クラスはシリアル化用に登録されます。そうしないと、基本クラスはネイティブモードでのシリアル化のために自動的に登録されません。シリアル化のために自動的に登録されたクラスのリストは、CamelSerializationProcessor.BASE_SERIALIZATION_CLASSES にあります。これを **false** に設定すると、ネイティブイメージのサイズを縮小することができます。JVM モードでは、ネイティブモードとの動作の一貫性を確立する場合を除き、このフラグを **true** に設定することは実際の利点がありません。	**boolean**	**false**
quarkus.camel.csimple.on-build-time-analysis-failure ビルド時にルート定義から CSimple 式を抽出できない場合の指示。	**org.apache.camel.quarkus.core.CamelConfig.FailureRemedy**	**warn**

設定プロパティー	タイプ	デフォルト
quarkus.camel.event-bridge.enabled Camel イベントから CDI イベントへのブリッジを有効にするかどうか。これにより、CDI オブザーバーを Camel イベント用に設定できます。たとえば、**org.apache.camel.quarkus.core.events**、**org.apache.camel.quarkus.main.events**、および **org.apache.camel.impl.event** パッケージに属するもの。この設定項目は、Camel イベント用に設定されたオブザーバーがアプリケーションに存在する場合にのみ効果があることに注意してください。	**boolean**	**true**
quarkus.camel.main.enabled **true** の場合、すべての **camel-main** 機能が有効になります。それ以外の場合、**camel-main** 機能は有効になりません。	**boolean**	**true**
quarkus.camel.main.shutdown.timeout **CamelMain#stop()** が完了するまで待機するタイムアウト時間 (ミリ秒の精度)	**java.time.Duration**	**PT3S**
quarkus.camel.main.arguments.on-unknown **CamelMain** が不明な引数に遭遇した際のアクション。fail: **CamelMain** の使用状況ステートメントを出力し、**RuntimeException** を出力します。ignore: 警告が解除し、アプリケーションは通常どおりに起動します。warn: **CamelMain** の使用状況ステートメントを出力しますが、アプリケーションが通常どおりに起動するのを許可します。	**org.apache.camel.quarkus.core.CamelConfig.FailureRemedy**	**warn**

ビルド時に修正される設定プロパティー。その他の設定プロパティーはすべて、ランタイム時にオーバーライドが可能です。

2.11. DIRECT

同じ Camel コンテキストから別のエンドポイントを同期的に呼び出します。

2.11.1. 含まれるもの

- Direct コンポーネント、URI 構文: **direct:name**

使用方法と設定の詳細については、上記リンクを参照してください。

2.11.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-direct</artifactId>
</dependency>
```

2.12. ELASTICSEARCH REST

REST API 経由で ElasticSearch を使用してリクエストを送信します。

2.12.1. 含まれるもの

- Elasticsearch Rest コンポーネント、URI 構文: **elasticsearch-rest:clusterName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.12.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-elasticsearch-rest</artifactId>
</dependency>
```

2.12.3. 用途

このエクステンションは、Quarkus ElasticSearch REST Client を活用します。

Quarkus 設定プロパティー を介して ElasticSearchを設定することを選択でき、**RestClient** は Camel ElasticSearch コンポーネントに自動配線されます。

または、Camel ElasticSearch コンポーネント/ エンドポイントオプションを介して ElasticSearch を設定できます。これを行うときは、以下に概説する方法の 1 つで自動配線を無効にする必要があります。

コンポーネントレベルでの自動配線の無効化。

```
camel.component.elasticsearch-rest.autowired-enabled = false
```

エンドポイントレベルでの自動配線の無効化。

```
from("direct:search")
    .to("elasticsearch-rest://elasticsearch?
hostAddresses=localhost:9200&operation=Search&indexName=index&autowiredEnabled=false")
```

自動配線をグローバルに無効にします。これにより、すべてのコンポーネントの自動配線が無効になることに注意してください。

```
camel.main.autowired-enabled = false
```

2.13. ファイル

ファイル読み取りおよび書き込みます。

2.13.1. 含まれるもの

- ファイルコンポーネント、URI 構文: **file:directoryName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.13.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-file</artifactId>
</dependency>
```

2.14. FTP

SFTP、FTP、または SFTP サーバーとの間でファイルをアップロードおよびダウンロードする

2.14.1. 含まれるもの

- FTP コンポーネント、URI 構文: **ftp:host:port/directoryName**
- FTPS コンポーネント、URI 構文: **ftps:host:port/directoryName**
- SFTP コンポーネント、URI 構文: **sftp:host:port/directoryName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.14.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-ftp</artifactId>
</dependency>
```

2.15. HL7

HL7 MLLP コーデックを使用して、HL7 (Health Care) モデルオブジェクトをマーシャリングおよびアンマーシャリングします。

2.15.1. 含まれるもの

- HL7 データフォーマット
- HL7 Terser 言語

使用方法と設定の詳細については、上記リンクを参照してください。

2.15.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-hl7</artifactId>
</dependency>
```

2.15.3. Camel Quarkus の制限

MLLP と TCP を使用する場合には、Netty が Hl7 MLLP リスナーを実行する唯一のサポートされる手段です。現在、GraalVM ネイティブサポートがないため、Mina はサポートされません。

HL7MLLPNettyEncoderFactory および **HL7MLLPNettyDecoderFactory** コーデックのオプションのサポートは、プロジェクト **pom.xml** の依存関係を **camel-quarkus-netty** に追加することで取得できます。

2.16. HTTP

Apache HTTP Client 4.x を使用して、外部の HTTP サーバーにリクエストを送信します。

2.16.1. 含まれるもの

- HTTP コンポーネント、URI 構文: **http://httpUri**
- HTTPS (Secure) コンポーネント、URI 構文: **https://httpUri**

使用方法と設定の詳細については、上記リンクを参照してください。

2.16.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-http</artifactId>
</dependency>
```

2.16.3. ネイティブモードの SSL

このエクステンションは、ネイティブモードでの SSL サポートを自動的に有効にします。したがって、自分で **quarkus.ssl.native=true** を **application.properties** に追加する必要はありません。「Quarkus SSL ガイド」も参照してください。

2.16.4. ネイティブモードの transferException オプション

ネイティブモードで**transferException**オプションを使用するには、オブジェクトのシリアル化のサポートを有効にする必要があります。詳細は、『**Developing Applications with Camel Extensions for Quarkus**』の「**Registering Classes for Serialization**」セクションを参照してください。

また、シリアル化する予定の例外クラスのシリアル化を有効にする必要があります。例:

```
@RegisterForReflection(targets = { IllegalStateException.class, MyCustomException.class },
serialization = true)
```

2.16.5. 追加の Camel Quarkus 設定

- デフォルト以外のエンコーディングを使用してアプリケーションを送受信することが予想される場合は、『**Developing Applications with Camel Extensions for Quarkus**』の「Character Encodings」セクション を確認してください。

2.17. JACKSON

Jackson を使用して、POJO を JSON にマーシャリングし、戻します。

2.17.1. 含まれるもの

- JSON Jackson データフォーマット

使用方法と設定の詳細については、上記リンクを参照してください。

2.17.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-jackson</artifactId>
</dependency>
```

2.18. AVRO JACKSON

Jackson を使用して、POJO を Avro にマーシャリングし、戻します。

2.18.1. 含まれるもの

- Avro Jackson データフォーマット

使用方法と設定の詳細については、上記リンクを参照してください。

2.18.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-jackson-avro</artifactId>
</dependency>
```

2.19. PROTOBUF JACKSON

Jackson を使用して、POJO を Protobuf にマーシャリングし、戻します。

2.19.1. 含まれるもの

- Protobuf Jackson データフォーマット

使用方法と設定の詳細については、上記リンクを参照してください。

2.19.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-jackson-protobuf</artifactId>
</dependency>
```

2.20. JACKSONXML

Jackson の XMLMapper エクステンションを使用して、XML ペイロードを POJO にアンマーシャリングし、戻します。

2.20.1. 含まれるもの

- JacksonXML データフォーマット

使用方法と設定の詳細については、上記リンクを参照してください。

2.20.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-jacksonxml</artifactId>
```

```
</dependency>
```

2.21. JIRA

JIRA 問題トラッカーと対話します。

2.21.1. 含まれるもの

- Jira コンポーネント、URI 構文: **jira:type**

使用方法と設定の詳細については、上記リンクを参照してください。

2.21.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-jira</artifactId>
</dependency>
```

2.21.3. ネイティブモードの SSL

このエクステンションは、ネイティブモードでの SSL サポートを自動的に有効にします。したがって、自分で **quarkus.ssl.native=true** を **application.properties** に追加する必要はありません。
「Quarkus SSL ガイド」も参照してください。

2.22. JMS

JMS Queue または Topic との間でメッセージを送受信します。

2.22.1. 含まれるもの

- JMS コンポーネント、URI 構文: **jms:destinationType:destinationName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.22.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-jms</artifactId>
</dependency>
```

2.22.3. 用途

2.22.3.1. org.w3c.dom.Nodeを使用したメッセージマッピング

Camel JMS コンポーネントは、**javax.jms.Message**および**org.apache.camel.Message**間のメッセージマッピングをサポートします。Camel メッセージ本文タイプ **org.w3c.dom.Node** を変換する場合は、**camel-quarkus-jaxp** エクステンションがクラスパスに存在することを確認する必要があります。

2.22.3.2. javax.jms.ObjectMessage のネイティブモードのサポート

JMS メッセージペイロードを**javax.jms.ObjectMessage**として送信する場合、シリアル化のために登録する関連クラスに**@RegisterForReflection (serialization = true)**でアノテーションを付ける必要があります。このエクステンションは、**quarkus.camel.native.reflection.serialization-enabled = true** を自動的に設定することに注意してください。詳細は、ネイティブモードのユーザーガイド を参照してください。

2.22.4. ネイティブモードの transferException オプション

ネイティブモードで**transferException**オプションを使用するには、オブジェクトのシリアル化のサポートを有効にする必要があります。詳細は、ネイティブモードのユーザーガイド を参照してください。

また、シリアル化する予定の例外クラスのシリアル化を有効にする必要があります。例:

```
@RegisterForReflection(targets = { IllegalStateException.class, MyCustomException.class },
serialization = true)
```

2.23. JSON PATH

JSON メッセージのボディーに対して、JsonPath 式を評価します。

2.23.1. 含まれるもの

- JsonPath 言語

使用方法と設定の詳細については、上記リンクを参照してください。

2.23.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-jsonpath</artifactId>
</dependency>
```

2.24. JTA

Java Transaction API (JTA) および Narayana トランザクションマネージャーを使用して、Camel ルートをトランザクションに含めます。

2.24.1. 含まれるもの

- JTA

使用方法と設定の詳細については、上記リンクを参照してください。

2.24.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-jta</artifactId>
</dependency>
```

2.24.3. 用途

このエクステンションは、ルーターで **transacted()** EIP を使用する必要がある場合に追加する必要があります。これは、Quarkus の narayana-jta エクステンションによって提供されるトランザクション機能を利用します。

トランザクションサポートの詳細は、「Quarkus Transaction guide」を参照してください。簡単な使用方法の場合:

```
from("direct:transaction")
  .transacted()
  .to("sql:INSERT INTO A TABLE ...?dataSource=ds1")
  .to("sql:INSERT INTO A TABLE ...?dataSource=ds2")
  .log("all data are in the ds1 and ds2")
```

さまざまなトランザクションポリシーのサポートが提供されます。

ポリシー	説明
PROPAGATION_MANDATORY	現在のトランザクションをサポートします。現在のトランザクションが存在しない場合は例外が発生します。
PROPAGATION_NEVER	現在のトランザクションをサポートしません。現在のトランザクションが存在する場合は例外が発生します。
PROPAGATION_NOT_SUPPORTED	現在のトランザクションはサポートせず、常に非トランザクションを実行します。
PROPAGATION_REQUIRED	現在のトランザクションをサポートします。存在しない場合は新しいトランザクションを作成します。
PROPAGATION_REQUIRES_NEW	新しいトランザクションを作成し、現在のトランザクションが存在する場合はそれを一時停止します。

ポリシー	説明
PROPAGATION_SUPPORTS	現在のトランザクションをサポートします。存在しない場合は、非トランザクションを実行します。

2.25. KAFKA

Apache Kafka ブローカーとの間でメッセージを送受信します。

2.25.1. 含まれるもの

- Kafka コンポーネント、URI 構文: **kafka:topic**

使用方法と設定の詳細については、上記リンクを参照してください。

2.25.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-kafka</artifactId>
</dependency>
```

2.25.3. 追加の Camel Quarkus 設定

設定プロパティー	タイプ	デフォルト
quarkus.camel.kafka.kubernetes-service-binding.merge-configuration **true** の場合、Quarkus Kubernetes Service Binding エクステンション (設定されている場合) によって検出された Kafka 設定プロパティーは、Camel Kafka コンポーネントまたはエンドポイントオプションを介して設定されたプロパティーとマージされます。**false** の場合、Quarkus Kubernetes Service Binding エクステンションによって検出された Kafka 設定プロパティーはすべて無視され、Kafka コンポーネント設定はすべて Camel によって駆動されます。	**boolean**	**true**

🔒 ビルド時に修正される設定プロパティー。その他の設定プロパティーはすべて、ランタイム時にオーバーライドが可能です。

2.26. KAMELET

ルートテンプレートを具体化する

2.26.1. 含まれるもの

- Kamelet コンポーネント、URI 構文: **kamelet:templateId/routeId**

使用方法と設定の詳細については、上記リンクを参照してください。

2.26.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-kamelet</artifactId>
</dependency>
```

2.26.3. 用途

2.26.3.1. ビルド時に Kamelets をプリロードします

このエクステンションを使用すると、**quarkus.camel.kamelet.identifiers** プロパティーを使用して、ビルド時に一連の Kamelet をプリロードできます。

2.26.4. 追加の Camel Quarkus 設定

設定プロパティー	タイプ	デフォルト
quarkus.camel.kamelet.identifiers ビルド時にプリロードする kamelets 識別子のリスト。個々の識別子は、関連する **org.apache.camel.model.RouteTemplateDefinition** id を設定するために使用されます。	**string**	

ビルド時に修正される設定プロパティー。その他の設定プロパティーはすべて、ランタイム時にオーバーライドが可能です。

2.27. LOG

基礎となるロギングメカニズムにメッセージをログとして記録します。

2.27.1. 含まれるもの

- Log コンポーネント、URI 構文: **log:loggerName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.27.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-log</artifactId>
</dependency>
```

2.28. MICROPROFILE HEALTH

Eclipse MicroProfile Health と Camel ヘルスチェックをブリッジングします。

2.28.1. 含まれるもの

- Microprofile Health

使用方法と設定の詳細については、上記リンクを参照してください。

2.28.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-microprofile-health</artifactId>
</dependency>
```

2.28.3. 用途

デフォルトでは、**AbstractHealthCheck** を拡張するクラスは liveness および readiness チェックの両方として登録されます。**isReadiness** メソッドを上書きして、この動作を制御できます。

アプリケーションによって提供されるチェックは自動的に検出され、Camel レジストリーにバインドされます。これらは、Quarkus ヘルスエンドポイント **/q/health/live** および **/q/health/ready** から利用できます。

カスタムの **HealthCheckRepository** 実装も提供でき、これらの実装も自動的に検出され、Camel レジストリーにバインドされます。

詳細は、「Quarkus health guide」を参照してください。

2.28.3.1. 提供されるヘルスチェック

一部のチェックはアプリケーションに自動的に登録されます。

2.28.3.1.1. Camel Context Health

Camel Context のステータスを検査して、ステータスが「Started」以外の場合にヘルスチェックのステータスを **DOWN** にします。

2.28.3.1.2. Camel Route Health

各ルートのステータスを検査して、いずれかのルートのステータスが「Started」以外の場合にヘルスチェックのステータスを **DOWN** にします。

2.28.4. 追加の Camel Quarkus 設定

設定プロパティー	タイプ	デフォルト
quarkus.camel.health.enabled Camel ヘルスチェックを有効にするかどうかを設定します	**boolean**	**true**

ビルド時に修正される設定プロパティー。その他の設定プロパティーはすべて、ランタイム時にオーバーライドが可能です。

2.29. MICROPROFILE METRICS

Camel ルートからメトリクスを公開します。

2.29.1. 含まれるもの

- MicroProfile Metrics コンポーネント、URI 構文: **microprofile-metrics:metricType:metricName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.29.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-microprofile-metrics</artifactId>
</dependency>
```

2.29.3. 用途

microprofile-metrics コンポーネントは Camel アプリケーションメトリクスのセットを自動的に公開します。これには以下が含まれます。

2.29.3.1. Camel Context メトリクス

メトリクス名	タイプ
camel.context.status **ServiceStatus** 列挙順序で表される Camel Context のステータス	ゲージ
camel.context.uptime Camel Context のアップタイム (ミリ秒単位)	ゲージ
camel.context.exchanges.completed.total 完了したエクスチェンジの合計数	カウンター
camel.context.exchanges.failed.total 失敗したエクスチェンジの合計数	カウンター
camel.context.exchanges.inflight.total インフライトエクスチェンジの合計数	ゲージ
camel.context.exchanges.total すべてのエクスチェンジの合計数	カウンター
camel.context.externalRedeliveries.total すべての外部再配信の合計数	カウンター
camel.context.failuresHandled.total 処理されたすべての障害の合計数	カウンター

2.29.3.2. Camel Route メトリクス

メトリクス名	タイプ
camel.route.count ルート数	ゲージ
camel.route.running.count 実行中のルート数	ゲージ
camel.route.exchanges.completed.total ルートの完了したエクスチェンジの合計数	カウンター

メトリクス名	タイプ
camel.route.exchanges.failed.total ルートの失敗したエクスチェンジの合計数	カウンター
camel.route.exchanges.inflight.total ルートのインフライトエクスチェンジの合計数	ゲージ
camel.route.exchanges.total ルートのすべてのエクスチェンジの合計数	カウンター
camel.route.externalRedeliveries.total ルートのすべての外部再配信の合計数	カウンター
camel.route.failuresHandled.total ルートの処理されたすべての障害の合計数	カウンター

すべてのメトリクスは、Camel Context の名前と、該当する場合にルートの ID でタグ付けされます。

Camel ルートで独自のカスタムメトリクスを生成することもできます。詳細は、microprofile-metrics コンポーネントのドキュメントを参照してください。

メトリクスはアプリケーションメトリクスとして Quarkus に公開され、http://localhost:8080/q/metrics/application で参照することができます。

2.29.4. 追加の Camel Quarkus 設定

設定プロパティー	タイプ	デフォルト
quarkus.camel.metrics.enable-route-policy ルート処理時間のメトリクスをキャプチャーするために MicroProfileMetricsRoutePolicyFactory を有効にするかどうかを設定します。	**boolean**	**true**
quarkus.camel.metrics.enable-message-history 個々のルートノード処理時間のメトリクスをキャプチャーするために MicroProfileMetricsMessageHistoryFactory を有効にするかどうかを設定します。設定されたルートノードの数によっては、大量のメトリクスが作成される可能性があります。したがって、このオプションはデフォルトで無効になります。	**boolean**	**false**

設定プロパティー	タイプ	デフォルト
quarkus.camel.metrics.enable-exchange-event-notifier エクスチェンジ処理時間のメトリクスをキャプチャーするために MicroProfileMetricsExchangeEventNotifier を有効にするかどうかを設定します。	**boolean**	**true**
quarkus.camel.metrics.enable-route-event-notifier ルートの合計数と実行中のルートの合計数のメトリクスをキャプチャーするために MicroProfileMetricsRouteEventNotifier を有効にするかどうかを設定します。	**boolean**	**true**
quarkus.camel.metrics.enable-camel-context-event-notifier ステータスやアップタイムなどの CamelContext に関するメトリックをキャプチャーするために MicroProfileMetricsCamelContextEventNotifier を有効にするかどうかを設定します。	**boolean**	**true**

ビルド時に修正される設定プロパティー。その他の設定プロパティーはすべて、ランタイム時にオーバーライドが可能です。

2.30. MLLP

MLLP プロトコルを使用して外部システムと通信します。

2.30.1. 含まれるもの

- MLLP コンポーネント、URI 構文: **mllp:hostname:port**

使用方法と設定の詳細については、上記リンクを参照してください。

2.30.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-mllp</artifactId>
</dependency>
```

2.30.3. 追加の Camel Quarkus 設定

2.31. MOCK

モックを使用してルートおよび仲介ルールをテストします。

2.31.1. 含まれるもの

- Mock コンポーネント、URI 構文: **mock:name**

使用方法と設定の詳細については、上記リンクを参照してください。

2.31.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-mock</artifactId>
</dependency>
```

2.31.3. 用途

テストで camel-mock 機能を使用するには、MockEndpoint インスタンスへのアクセスを取得する必要があります。

CDI の注入は、インスタンスへのアクセスに使用できます (Quarkus ドキュメント を参照してください)。**@Inject** アノテーションを使用して camelContext をテストに注入できます。その後、Camel コンテキストを使用してモックエンドポイントを取得できます。以下の例を参照してください。

```
import javax.inject.Inject;

import org.apache.camel.CamelContext;
import org.apache.camel.ProducerTemplate;
import org.apache.camel.component.mock.MockEndpoint;
import org.junit.jupiter.api.Test;

import io.quarkus.test.junit.QuarkusTest;

@QuarkusTest
public class MockJvmTest {

    @Inject
    CamelContext camelContext;

    @Inject
    ProducerTemplate producerTemplate;

    @Test
    public void test() throws InterruptedException {

        producerTemplate.sendBody("direct:start", "Hello World");

        MockEndpoint mockEndpoint = camelContext.getEndpoint("mock:result", MockEndpoint.class);
```

```
        mockEndpoint.expectedBodiesReceived("Hello World");

        mockEndpoint.assertIsSatisfied();
    }
}
```

サンプルテストに使用するルート:

```
import javax.enterprise.context.ApplicationScoped;

import org.apache.camel.builder.RouteBuilder;

@ApplicationScoped
public class MockRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("direct:start").to("mock:result");
    }
}
```

2.31.4. Camel Quarkus の制限

(「使用法」で説明した) CDI Bean の注入は、ネイティブモードでは機能しません。

ネイティブモードでは、テストとテスト中のアプリケーションが 2 つの異なるプロセスで実行され、それらの間でモック Bean を共有することはできません (Quarkus ドキュメント を参照)。

2.32. MONGODB

MongoDB ドキュメントおよびコレクションの操作を実行します。

2.32.1. 含まれるもの

- MongoDB コンポーネント、URI 構文: **mongodb:connectionBean**

使用方法と設定の詳細については、上記リンクを参照してください。

2.32.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-mongodb</artifactId>
</dependency>
```

2.32.3. 追加の Camel Quarkus 設定

エクステンションは Quarkus MongoDB Client エクステンションを活用します。Mongo クライアントは、Quarkus MongoDB Client の 設定オプション を使用して設定できます。

Camel Quarkus MongoDB エクステンションは、**camelMongoClient** という名前の MongoDB クライアント Bean を自動的に登録します。これは、mongodb エンドポイント URI の **connectionBean** パスパラメーターで参照できます。以下は例になります。

```
from("direct:start")
.to("mongodb:camelMongoClient?database=myDb&collection=myCollection&operation=findAll")
```

アプリケーションが複数の MongoDB サーバーと連携する必要がある場合は、「Quarkus MongoDB extension client injection」で説明するように、「特定の名前の」クライアントを作成し、クライアントおよび関連する設定を注入することで、ルートで参照できます。以下は例になります。

```
//application.properties
quarkus.mongodb.mongoClient1.connection-string = mongodb://root:example@localhost:27017/
```

```
//Routes.java

    @ApplicationScoped
    public class Routes extends RouteBuilder {
        @Inject
        @MongoClientName("mongoClient1")
        MongoClient mongoClient1;

        @Override
        public void configure() throws Exception {
            from("direct:defaultServer")
                .to("mongodb:camelMongoClient?
database=myDb&collection=myCollection&operation=findAll")

            from("direct:otherServer")
                .to("mongodb:mongoClient1?
database=myOtherDb&collection=myOtherCollection&operation=findAll");
        }
    }
```

指定されたクライアントを使用する場合、「デフォルトの」**camelMongoClient** Bean は引き続き生成されます。詳細は、複数の MongoDB クライアント に関する Quarkus ドキュメントを参照してください。

2.33. NETTY

Netty 4.x で TCP または UDP を使用するソケットレベルのネットワーク。

2.33.1. 含まれるもの

- Netty コンポーネント、URI 構文: **netty:protocol://host:port**

使用方法と設定の詳細については、上記リンクを参照してください。

2.33.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-netty</artifactId>
</dependency>
```

2.34. OPENAPI JAVA

Camel REST DSLL で定義された OpenAPI リソースを公開する

2.34.1. 含まれるもの

- Openapi Java

使用方法と設定の詳細については、上記リンクを参照してください。

2.34.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-openapi-java</artifactId>
</dependency>
```

2.34.3. Camel Quarkus の制限

apiContextIdListing 設定オプションはサポートされません。複数の**CamelContext**はサポートされておらず、Quarkus アプリケーションはスタンドアロンで実行されるため、特定の**CamelContext**の OpenApi 仕様を解決しようとすることが役立つシナリオはありません。また、JMX (ネイティブモードではサポートされていません) を必要とする追加のオーバーヘッドと、XML を処理するための追加の Camel Quarkus エクステンションも導入されています。

2.35. PLATFORM HTTP

このエクステンションにより、HTTP リクエストを使用するために HTTP エンドポイントを作成できます。

これは、**quarkus-vertx-web** エクステンションによって提供される Eclipse Vert.x Web サービスに加えて構築されます。

2.35.1. 含まれるもの

- Platform HTTP コンポーネント、URI 構文: **platform-http:path**

使用方法と設定の詳細については、上記リンクを参照してください。

2.35.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-platform-http</artifactId>
</dependency>
```

2.35.3. 用途

2.35.3.1. 基本的な使用方法

/hello エンドポイントですべての HTTP メソッドを提供します。

```
from("platform-http:/hello").setBody(simple("Hello ${header.name}"));
```

/hello エンドポイントで GET リクエストのみを提供します。

```
from("platform-http:/hello?httpMethodRestrict=GET").setBody(simple("Hello ${header.name}"));
```

2.35.3.2. Camel REST DSL 経由の platform-http の使用

platform-http コンポーネントで Camel REST DSL を使用できるようにするには、 **camel-quarkus-platform-http** に加えて **camel-quarkus-rest** を **pom.xml** に追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-rest</artifactId>
</dependency>
```

その後、Camel REST DSL を使用できます。

```
rest()
    .get("/my-get-endpoint")
        .route()
        .setBody(constant("Hello from /my-get-endpoint"))
        .endRest()
    .post("/my-post-endpoint")
        .route()
        .setBody(constant("Hello from /my-post-endpoint"))
        .endRest();
```

2.35.3.3. multipart/form-data ファイルのアップロードの処理

ホワイトリストに登録して、アップロードを特定のファイル拡張子に制限することができます。

```
from("platform-http:/upload/multipart?fileNameExtWhitelist=html,txt&httpMethodRestrict=POST")
    .to("log:multipart")
    .process(e -> {
        final AttachmentMessage am = e.getMessage(AttachmentMessage.class);
        if (am.hasAttachments()) {
            am.getAttachments().forEach((fileName, dataHandler) -> {
```

```
            try (InputStream in = dataHandler.getInputStream()) {
                // do something with the input stream
            } catch (IOException ioe) {
                throw new RuntimeException(ioe);
            }
        });
    }
});
```

Quarkus ドキュメント で **quarkus.http.body.*** 設定オプション (特に次の項目) も確認してください。**quarkus.http.body.handle-file-uploads**、**quarkus.http.body.uploads-directory** および **quarkus.http.body.delete-uploaded-files-on-end**。

2.35.4. 追加の Camel Quarkus 設定

2.35.4.1. プラットフォーム HTTP サーバー設定

プラットフォーム HTTP サーバーの設定は Quarkus によって管理されます。設定オプションの完全なリストについては、Quarkus HTTP 設定ガイド を参照してください。

Platform HTTP サーバーの SSL を設定するには、SSL ガイドを使用した安全な接続 に従ってください。**SSLContextParameters** を使用した SSL 用のサーバーの設定は現在サポートされていないことに注意してください。

2.35.4.2. 文字エンコーディング

- デフォルト以外のエンコーディングを使用してアプリケーションを送受信することが予想される場合は、『 **Developing Applications with Camel Extensions for Quarkus**』の 「Character Encodings」セクション を確認してください。

2.36. REST

REST サービスおよび OpenAPI Specification を公開するか、外部の REST サービスを呼び出します。

2.36.1. 含まれるもの

- REST コンポーネント、URI 構文: **rest:method:path:uriTemplate**
- REST API コンポーネント、URI 構文: **rest-api:path/contextIdPattern**

使用方法と設定の詳細については、上記リンクを参照してください。

2.36.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-rest</artifactId>
</dependency>
```

2.36.3. 追加の Camel Quarkus 設定

このエクステンションは、Platform HTTP エクステンションに依存し、REST トランスポートを提供するコンポーネントとして設定します。

2.36.3.1. platform-http 付きの特殊文字を含むパスパラメーター

platform-http REST トランスポートを使用する場合、一部の文字はパスパラメーター名内で許可されません。これには、- および $ 文字が含まれます。

以下の例の REST **/dashed/param** ルートを正しく機能させるには、システムプロパティーを **io.vertx.web.route.param.extended-pattern=true** にする必要があります。

```
import org.apache.camel.builder.RouteBuilder;

public class CamelRoute extends RouteBuilder {

    @Override
    public void configure() {
        rest("/api")
            // Dash '-' is not allowed by default
            .get("/dashed/param/{my-param}")
            .route()
                .setBody(constant("Hello World"))
            .endRest()

            // The non-dashed path parameter works by default
            .get("/undashed/param/{myParam}")
            .route()
                .setBody(constant("Hello World"))
            .endRest();
    }
}
```

Vert.x Web ドキュメント には、これに関するいくつかの背景があります。

2.36.3.2. 代替 REST トランスポートプロバイダーの設定

netty-http や **servlet** 等の別の REST トランスポートプロバイダーを使用するには、それぞれのエクステンションを依存関係としてプロジェクトへの追加し、**RouteBuilder** でプロバイダーを設定する必要があります。たとえば、**servlet** の場合、**org.apache.camel.quarkus:camel-quarkus-servlet** 依存関係を追加し、プロバイダーを次のように設定する必要があります。

```
import org.apache.camel.builder.RouteBuilder;

public class CamelRoute extends RouteBuilder {

    @Override
    public void configure() {
        restConfiguration()
                .component("servlet");
        ...
    }
}
```

2.37. SALESFORCE

Java DTO を使用して Salesforce と通信します。

2.37.1. 含まれるもの

- Salesforce コンポーネント、URI 構文: **salesforce:operationName:topicName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.37.2. Maven コーディネート

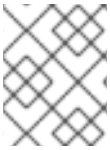
code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-salesforce</artifactId>
</dependency>
```

2.37.3. 用途

2.37.3.1. salesforce-maven-plugin を使用した Salesforce DTO の生成



注記

camel-salesforce-maven-plugin は、コミュニティーのサポートによってのみカバーされます。

プロジェクトの Salesforce DTO を生成するには、**salesforce-maven-plugin** を使用します。以下のサンプルコードスニペットは、**Account** オブジェクトの単一の DTO を作成します。

```
<plugin>
  <groupId>org.apache.camel.maven</groupId>
  <artifactId>camel-salesforce-maven-plugin</artifactId>
  <version>3.11.1</version>
  <executions>
    <execution>
      <goals>
        <goal>generate</goal>
      </goals>
      <configuration>
        <clientId>${env.SALESFORCE_CLIENTID}</clientId>
        <clientSecret>${env.SALESFORCE_CLIENTSECRET}</clientSecret>
        <userName>${env.SALESFORCE_USERNAME}</userName>
        <password>${env.SALESFORCE_PASSWORD}</password>
        <loginUrl>https://login.salesforce.com</loginUrl>

<packageName>org.apache.camel.quarkus.component.salesforce.generated</packageName>
        <outputDirectory>src/main/java</outputDirectory>
        <includes>
          <include>Account</include>
```

```
            </includes>
          </configuration>
        </execution>
      </executions>
    </plugin>
```

2.37.3.2. ネイティブモードで反映するために追加の Salesforce クラスを登録する

ネイティブモードの場合は、リフレクション用にいくつかの追加クラスを登録する必要があります。

1. パッケージ **org.apache.camel.component.salesforce.api.dto** のクラス
2. **camel-salesforce-maven-plugin** によって生成された DTO クラス

これを行うには、次の設定プロパティーを **application.properties** に追加します。**org.my.custom.dto.package** をカスタム DTO パッケージに置き換えます (該当する場合は、削除できます)。

```
quarkus.camel.native.reflection.include-
patterns=org.apache.camel.component.salesforce.api.dto.*,org.my.custom.dto.package.*
```

2.37.4. ネイティブモードの SSL

このエクステンションは、ネイティブモードでの SSL サポートを自動的に有効にします。したがって、自分で **quarkus.ssl.native=true** を **application.properties** に追加する必要はありません。「Quarkus SSL ガイド」も参照してください。

2.38. XQUERY

XQuery および Saxon を使用して XML ペイロードをクエリーまたは変換します。

2.38.1. 含まれるもの

- XQuery コンポーネント、URI 構文: **xquery:resourceUri**
- XQuery 言語

使用方法と設定の詳細については、上記リンクを参照してください。

2.38.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-saxon</artifactId>
</dependency>
```

2.38.3. 追加の Camel Quarkus 設定

このコンポーネントは、クラスパスから XQuery 定義をロードできます。ネイティブモードでも機能させるには、**quarkus.native.resources.includes** プロパティーを使用して、クエリをネイティブ実行可能ファイルに明示的に埋め込む必要があります。

たとえば、以下の 2 つのルートは、それぞれ **myxquery.txt** および **another-xquery.txt** という名前の 2 つのクラスパスリソースから XQuery スクリプトをロードします。

```
from("direct:start").transform().xquery("resource:classpath:myxquery.txt", String.class);
from("direct:start").to("xquery:another-xquery.txt");
```

これら (**.txt** ファイルに保存されている可能性のある他のクエリー) をネイティブイメージに含めるには、**application.properties** ファイルに次のようなものを追加する必要があります。

```
quarkus.native.resources.includes = *.txt
```

2.39. SEDA

同じ JVM の Camel コンテキストから別のエンドポイントを非同期に呼び出します。

2.39.1. 含まれるもの

- SEDA コンポーネント、URI 構文: **seda:name**

使用方法と設定の詳細については、上記リンクを参照してください。

2.39.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
    <groupId>org.apache.camel.quarkus</groupId>
    <artifactId>camel-quarkus-seda</artifactId>
</dependency>
```

2.40. SOAP DATAFORMAT

Java オブジェクトを SOAP メッセージにマーシャリングし、戻します。

2.40.1. 含まれるもの

- SOAP データフォーマット

使用方法と設定の詳細については、上記リンクを参照してください。

2.40.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-soap</artifactId>
</dependency>
```

2.41. SQL

SQL クエリを実行します。

2.41.1. 含まれるもの

- SQL コンポーネント、URI 構文: **sql:query**
- SQL Stored Procedure コンポーネント、URI 構文: **sql-stored:template**

使用方法と設定の詳細については、上記リンクを参照してください。

2.41.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-sql</artifactId>
</dependency>
```

2.41.3. 追加の Camel Quarkus 設定

2.41.3.1. データソースの設定

このエクステンションは、**DataSource** のサポートに Quarkus Agroal を活用します。**DataSource** の設定は、設定プロパティーを介して実行できます。

```
quarkus.datasource.db-kind=postgresql
quarkus.datasource.username=your-username
quarkus.datasource.password=your-password
quarkus.datasource.jdbc.url=jdbc:postgresql://localhost:5432/your-database
quarkus.datasource.jdbc.max-size=16
```

Camel SQL コンポーネントは、レジストリーから **DataSource** Bean を自動的に解決します。複数のデータソースを設定する場合、URI オプション **datasource** または **dataSourceRef** を使用して、SQL エンドポイントで使用するデータソースを指定できます。詳細については、SQL コンポーネントのドキュメントを参照してください。

2.41.3.1.1. Quarkus Dev Services によるゼロ設定

開発モードとテストモードでは、Configuration Free Databases を利用できます。Camel SQL コンポーネントは、選択した JDBC ドライバータイプに一致するデータベースのローカルコンテナー化インスタンスを指す **DataSource** を使用するように自動的に設定されます。

2.41.3.2. SQL スクリプト

クラスパスからスクリプトファイルを参照するために **sql** または **sql-stored** エンドポイントを設定する場合は、以下の設定プロパティーを設定して、それらがネイティブモードで利用できるようにします。

```
quarkus.native.resources.includes = queries.sql, sql/*.sql
```

2.41.3.3. SQL アグリゲーター

ネイティブモードのエクスチェンジに、シリアル化のために自動的に登録されないオブジェクトが含まれている場合 (ドキュメントを参照)、手動で登録する必要があります (ドキュメント を参照)。

2.42. TIMER

java.util.Timer を使用して指定した間隔でメッセージを生成します。

2.42.1. 含まれるもの

- Timer コンポーネント、URI 構文: **timer:timerName**

使用方法と設定の詳細については、上記リンクを参照してください。

2.42.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
  <artifactId>camel-quarkus-timer</artifactId>
</dependency>
```

2.43. XPATH

XML ペイロードに対して XPath 式を評価します。

2.43.1. 含まれるもの

- XPath 言語

使用方法と設定の詳細については、上記リンクを参照してください。

2.43.2. Maven コーディネート

code.quarkus.redhat.com でこのエクステンションを使用して新しいプロジェクトの作成

または、既存のプロジェクトに座標を追加します。

```
<dependency>
  <groupId>org.apache.camel.quarkus</groupId>
```

```
    <artifactId>camel-quarkus-xpath</artifactId>
</dependency>
```

2.43.3. 追加の Camel Quarkus 設定

このコンポーネントは、クラスパスリソースから xpath 式をロードできます。ネイティブモードでも機能させるには、**quarkus.native.resources.includes** プロパティーを使用して、式ファイルをネイティブ実行可能ファイルに明示的に埋め込む必要があります。

たとえば、以下のルートは、**myxpath.txt** という名前のクラスパスリソースから XPath 表現を読み込みます。

```
from("direct:start").transform().xpath("resource:classpath:myxpath.txt");
```

これら (**.txt** ファイルに保存されている可能性のある他の式) をネイティブイメージに含めるには、**application.properties** ファイルに次のようなものを追加する必要があります。

```
quarkus.native.resources.includes = *.txt
```