Red Hat Integration 2021.Q3

Camel Extensions for Quarkus

SUBTITLE

Last Updated: 2021-08-05

Red Hat Integration 2021.Q3 Camel Extensions for Quarkus

SUBTITLE

Legal Notice

Abstract

ABSTRACT

Table of Contents

PREFACE

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

CHAPTER 1. EXTENSIONS OVERVIEW

1.1. SUPPORT LEVEL DEFINITIONS

New features, services, and components go through a number of support levels before inclusion in Camel Quarkus as fully supported for production use. This is to ensure the right balance between providing the enterprise stability expected of our offerings with the need to allow our customers and partners to experiment with new Camel Quarkus technologies while providing feedback to help guide future development activities.

Table 1.1. Camel Extensions for Quarkus support levels

Type	Description
Community Support	As part of Red Hat's commitment to upstream first, integration of new extensions into our Camel Quarkus distribution begins in the upstream community. While these Camel Quarkus extensions have been tested and documented upstream, we have not reviewed the maturity of these extensions and they may not be formally supported by Red Hat in future product releases.
Technology Preview	Technology Preview features provide early access to upcoming product innovations, enabling you to test functionality and provide feedback during the development process. However, these features are not fully supported under Red Hat Subscription Level Agreements, may not be functionally complete, and are not intended for production use. As Red Hat considers making future iterations of Technology Preview features generally available, we will attempt to resolve any issues that customers experience when using these features.  **NOTE** Technology Preview support is limited to JVM mode in this release.
Production Support	Production Support extensions are shipped in a formal Red Hat release and are fully supported. There are no documentation gaps and extensions have been tested on all supported configurations.

1.2. SUPPORTED EXTENSIONS

There are 34 extensions.

Table 1.2. Camel Extensions for Quarkus Support Matrix

Extension	Artifact	Support Level	Description
AWS 2 DynamoDB	camel-quarkus-aws2-ddb	Technology Preview	Store and retrieve data from AWS DynamoDB service or receive messages from AWS DynamoDB Stream using AWS SDK version 2.x.
AWS 2 Kinesis	camel-quarkus-aws2-kinesis	Technology Preview	Consume and produce records from AWS Kinesis Streams using AWS SDK version 2.x.
AWS 2 Lambda	camel-quarkus-aws2-lambda	Technology Preview	Manage and invoke AWS Lambda functions using AWS SDK version 2.x.
AWS 2 S3 Storage Service	camel-quarkus-aws2-s3	Technology Preview	Store and retrieve objects from AWS S3 Storage Service using AWS SDK version 2.x.
AWS 2 Simple Notification System (SNS)	camel-quarkus-aws2-sns	Technology Preview	Send messages to an AWS Simple Notification Topic using AWS SDK version 2.x.
AWS 2 Simple Queue Service (SQS)	camel-quarkus-aws2-sqs	Technology Preview	Sending and receive messages to/from AWS SQS service using AWS SDK version 2.x.
Bean	camel-quarkus-bean	Technology Preview	Invoke methods of Java beans
Core	camel-quarkus-core	Technology Preview	Camel core functionality and basic Camel languages: Constant, ExchangeProperty, Header, Ref, Ref, Simple and Tokeinze
Direct	camel-quarkus-direct	Technology Preview	Call another endpoint from the same Camel Context synchronously.
Elasticsearch Rest	camel-quarkus-elasticsearch-rest	Technology Preview	Send requests to with an ElasticSearch via REST API.

Extension	Artifact	Support Level	Description
File	camel-quarkus-file	Technology Preview	Read and write files.
FTP	camel-quarkus-ftp	Technology Preview	Upload and download files to/from FTP or SFTP servers.
HTTP	camel-quarkus-http	Technology Preview	Send requests to external HTTP servers using Apache HTTP Client 4.x.
Jira	camel-quarkus-jira	Technology Preview	Interact with JIRA issue tracker.
JMS	camel-quarkus-jms	Technology Preview	Send and receive messages to/from a JMS Queue or Topic.
JTA	camel-quarkus-jta	Technology Preview	Enclose Camel routes in the transactions using Java Transaction API (JTA) and Narayana transaction manager
Kafka	camel-quarkus-kafka	Technology Preview	Sent and receive messages to/from an Apache Kafka broker.
Kamelet	camel-quarkus-kamelet	Technology Preview	The Kamelet Component provides support for interacting with the Camel Route Template engine.
Log	camel-quarkus-log	Technology Preview	Log messages to the underlying logging mechanism.
Main	camel-quarkus-main	Technology Preview	Bootstrap Camel using Camel Main which brings advanced auto-configuration capabilities and integration with Quarkus Command Mode
MicroProfile Health	camel-quarkus-microprofile-health	Technology Preview	Bridging Eclipse MicroProfile Health with Camel health checks.

Extension	Artifact	Support Level	Description
MicroProfile Metrics	camel-quarkus-microprofile-metrics	Technology Preview	Expose metrics from Camel routes.
MLLP	camel-quarkus-mllp	Technology Preview	Communicate with external systems using the MLLP protocol.
Mock	camel-quarkus-mock	Technology Preview	Test routes and mediation rules using mocks.
MongoDB	camel-quarkus-mongodb	Technology Preview	Perform operations on MongoDB documents and collections.
Netty	camel-quarkus-netty	Technology Preview	Socket level networking using TCP or UDP with the Netty 4.x.
Platform HTTP	camel-quarkus-platform-http	Technology Preview	Expose HTTP endpoints using the HTTP server available in the current platform.
Rest	camel-quarkus-rest	Technology Preview	Expose REST services and their OpenAPI Specification or call external REST services.
Salesforce	camel-quarkus-salesforce	Technology Preview	Communicate with Salesforce using Java DTOs.
XQuery	camel-quarkus-saxon	Technology Preview	Query and/or transform XML payloads using XQuery and Saxon.
SEDA	camel-quarkus-seda	Technology Preview	Asynchronously call another endpoint from any Camel Context in the same JVM.
SQL	camel-quarkus-sql	Technology Preview	Perform SQL queries using Spring JDBC.

Extension	Artifact	Support Level	Description
Timer	camel-quarkus-timer	Technology Preview	Generate messages in specified intervals using java.util.Timer.
XPath	camel-quarkus-xpath	Technology Preview	Evaluate an XPath expression against an XML payload.

1.3. SUPPORTED DATA FORMATS

There are 8 data formats.

Table 1.3. Camel Extensions for Quarkus Support Matrix

Extension	Artifact	Support Level	Description
Avro	camel-quarkus-avro	Technology Preview	Serialize and deserialize messages using Apache Avro binary data format.
Avro Jackson	camel-quarkus-jackson-avro	Technology Preview	Marshal POJOs to Avro and back using Jackson.
Bindy	camel-quarkus-bindy	Technology Preview	Marshal and unmarshal between POJOs and Comma separated values (CSV) format using Camel Bindy Marshal and unmarshal between POJOs and fixed field length format using Camel Bindy Marshal and unmarshal between POJOs and key-value pair (KVP) format using Camel Bindy
HL7	camel-quarkus-hl7	Technology Preview	Marshal and unmarshal HL7 (Health Care) model objects using the HL7 MLLP codec.
Jackson	camel-quarkus-jackson	Technology Preview	Marshal POJOs to JSON and back using Jackson

Extension	Artifact	Support Level	Description
JacksonXML	camel-quarkus-jacksonxml	Technology Preview	Unmarshal a XML payloads to POJOs and back using XMLMapper extension of Jackson.
Protobuf Jackson	camel-quarkus-jackson-protobuf	Technology Preview	Marshal POJOs to Protobuf and back using Jackson.
SOAP dataformat	camel-quarkus-soap	Technology Preview	Marshal Java objects to SOAP messages and back.

1.4. SUPPORTED LANGUAGES

There are 8 languages.

Table 1.4. Camel Extensions for Quarkus Support Matrix

Extension	Artifact	Support Level	Description
Constant	camel-quarkus-core	Technology Preview	A fixed value set only once during the route startup.
ExchangeProperty	camel-quarkus-core	Technology Preview	Get the value of named Camel Exchange property.
File	camel-quarkus-core	Technology Preview	For expressions and predicates using the file/simple language.
Header	camel-quarkus-core	Technology Preview	Get the value of the named Camel Message header.
Ref	camel-quarkus-core	Technology Preview	Look up an expression in the Camel Registry and evaluate it.
Simple	camel-quarkus-core	Technology Preview	Evaluate Camel's built-in Simple language expression against the Camel Exchange.

Extension	Artifact	Support Level	Description
Tokenize	camel-quarkus-core	Technology Preview	Tokenize text payloads using the specified delimiter patterns.
JSON Path	camel-quarkus-jsonpath	Technology Preview	Evaluate a JsonPath expression against a JSON message body.

CHAPTER 2. EXTENSIONS REFERENCE

This chapter provides reference information on the Camel Quarkus extensions.

IMPORTANT

This Technology Preview release includes a targeted subset of the available Camel Quarkus extensions. Additional extensions will be added to our Camel Quarkus distribution in future releases.

2.1. AVRO

Serialize and deserialize messages using Apache Avro binary data format.

2.1.1. What's inside

- Avro data format

Please refer to the above link for usage and configuration details.

2.1.2. Maven coordinates

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

2.1.3. Additional Camel Quarkus configuration

Beyond standard usages known from vanilla Camel, Camel Quarkus adds the possibility to parse the Avro schema at build time both in JVM and Native mode via the **@BuildTimeAvroDataFormat** annotation.

For instance below, in the first step the **user.avsc** schema resource is parsed at build time. In the second step, an AvroDataFormat instance using the previously parsed schema is injected in the **buildTimeAvroDataFormat** field at runtime. At the end of the day, the injected data format is used from the **configure()** method in order to marshal an incoming message.

```
@BuildTimeAvroDataFormat("user.avsc")
AvroDataFormat buildTimeAvroDataFormat;

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

2.2. AWS 2 DYNAMODB

Store and retrieve data from AWS DynamoDB service or receive messages from AWS DynamoDB Stream using AWS SDK version 2.x.

2.2.1. What's inside

- AWS 2 DynamoDB component, URI syntax: **aws2-ddb:tableName**
- AWS 2 DynamoDB Streams component, URI syntax: **aws2-ddbstream:tableName**

Please refer to the above links for usage and configuration details.

2.2.2. Maven coordinates

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

2.2.3. SSL in native mode

This extension auto-enables SSL support in native mode. Hence you do not need to add **quarkus.ssl.native=true** to your **application.properties** yourself. See also Quarkus SSL guide.

2.3. AWS 2 KINESIS

Consume and produce records from AWS Kinesis Streams using AWS SDK version 2.x.

2.3.1. What's inside

- AWS 2 Kinesis component, URI syntax: **aws2-kinesis:streamName**
- AWS 2 Kinesis Firehose component , URI syntax: **aws2-kinesis-firehose:streamName**

Please refer to the above links for usage and configuration details.

2.3.2. Maven coordinates

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

2.3.3. SSL in native mode

This extension auto-enables SSL support in native mode. Hence you do not need to add **quarkus.ssl.native=true** to your **application.properties** yourself. See also Quarkus SSL guide.

2.4. AWS 2 LAMBDA

Manage and invoke AWS Lambda functions using AWS SDK version 2.x.

2.4.1. What's inside

- AWS 2 Lambda component, URI syntax: **aws2-lambda:function**

Please refer to the above link for usage and configuration details.

2.4.2. Maven coordinates

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

2.4.3. SSL in native mode

This extension auto-enables SSL support in native mode. Hence you do not need to add **quarkus.ssl.native=true** to your **application.properties** yourself. See also Quarkus SSL guide.

2.5. AWS 2 S3 STORAGE SERVICE

Store and retrieve objects from AWS S3 Storage Service using AWS SDK version 2.x.

2.5.1. What's inside

- AWS 2 S3 Storage Service component, URI syntax: **aws2-s3://bucketNameOrArn**

Please refer to the above link for usage and configuration details.

2.5.2. Maven coordinates

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

2.5.3. SSL in native mode

This extension auto-enables SSL support in native mode. Hence you do not need to add **quarkus.ssl.native=true** to your **application.properties** yourself. See also Quarkus SSL guide.

2.6. AWS 2 SIMPLE NOTIFICATION SYSTEM (SNS)

Send messages to an AWS Simple Notification Topic using AWS SDK version 2.x.

2.6.1. What's inside

- AWS 2 Simple Notification System (SNS) component , URI syntax: **aws2-sns:topicNameOrArn**

Please refer to the above link for usage and configuration details.

2.6.2. Maven coordinates

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

```
</dependency>
```

2.6.3. SSL in native mode

This extension auto-enables SSL support in native mode. Hence you do not need to add **quarkus.ssl.native=true** to your **application.properties** yourself. See also Quarkus SSL guide.

2.7. AWS 2 SIMPLE QUEUE SERVICE (SQS)

Sending and receive messages to/from AWS SQS service using AWS SDK version 2.x.

2.7.1. What's inside

- AWS 2 Simple Queue Service (SQS) component, URI syntax: **aws2-sqs:queueNameOrArn**

Please refer to the above link for usage and configuration details.

2.7.2. Maven coordinates

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

2.7.3. SSL in native mode

This extension auto-enables SSL support in native mode. Hence you do not need to add **quarkus.ssl.native=true** to your **application.properties** yourself. See also Quarkus SSL guide.

2.8. BEAN

Invoke methods of Java beans

2.8.1. What's inside

- Bean component, URI syntax: **bean:beanName**
- Bean method language
- Class component, URI syntax: **class:beanName**

Please refer to the above links for usage and configuration details.

2.8.2. Maven coordinates

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

2.8.3. Usage

Except for invoking methods of beans available in Camel registry, Bean component and Bean method language can also invoke Quarkus CDI beans.

2.9. BINDY

Marshal and unmarshal between POJOs and Comma separated values (CSV) format using Camel Bindy Marshal and unmarshal between POJOs and fixed field length format using Camel Bindy Marshal and unmarshal between POJOs and key-value pair (KVP) format using Camel Bindy

2.9.1. What's inside

- Bindy CSV data format
- Bindy Fixed Length data format
- Bindy Key Value Pair data format

Please refer to the above links for usage and configuration details.

2.9.2. Maven coordinates

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

2.9.3. Camel Quarkus limitations

When using camel-quarkus-bindy in native mode, only the build machine's default locale is supported.

For instance, on build machines with french default locale, the code below:

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

formats numbers the arabic way in JVM mode as expected. However, it formats numbers the french way in native mode.

2.10. CORE

Camel core functionality and basic Camel languages/ Constant, ExchangeProperty, Header, Ref, Ref, Simple and Tokeinze

2.10.1. What's inside

- Constant language
- ExchangeProperty language
- File language
- Header language

- Ref language
- Simple language
- Tokenize language

Please refer to the above links for usage and configuration details.

2.10.2. Maven coordinates

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

2.10.3. Camel Quarkus limitations

2.10.3.1. Camel annotations

The following Camel annotations are not supported in this version of Camel Quarkus:

- **@org.apache.camel.EndpointInject**
- **@org.apache.camel.Produce**
- **@org.apache.camel.Consume**

2.10.4. Additional Camel Quarkus configuration

2.10.4.1. Simple language

2.10.4.1.1. Using the OGNL notation

When using the OGNL notation from the simple language, the **camel-quarkus-bean** extension should be used.

For instance, the expression below is accessing the **getAddress()** method on the message body of type **Client**.

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

In such a situation, one should take an additional dependency on the camel-quarkus-bean extension as described here. Note that in native mode, some classes may need to be registered for reflection. In the example above, the **Client** class needs to be registered for reflection.

2.10.4.1.2. Using dynamic type resolution in native mode

When dynamically resolving a type from simple expressions like **${mandatoryBodyAs(TYPE)}**, **${type:package.Enum.CONSTANT}** or **${body} is TYPE**, it may be needed to register some classes for reflection manually.

For instance, the simple expressions below is dynamically resolving the type **java.nio.ByteBuffer** at runtime:

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

As such, the class **java.nio.ByteBuffer** needs to be registered for reflection.

2.10.4.1.3. Using the simple language with classpath resources in native mode

Beyond standard usages, a trick is needed when using the simple language with classpath resources in native mode. In such a situation, one needs to explicitly embed the resources in the native executable by specifying the **include-patterns** option.

For instance, the route below would load a simple script from a classpath resource named *mysimple.txt*:

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

In order to work in native mode the **include-patterns** configuration should be set. For instance, in the **application.properties** file as below :

```
quarkus.camel.native.resources.include-patterns = *.txt
```

Configuration property	Type	Default
quarkus.camel.bootstrap.enabled When set to true, the **CamelRuntime** will be started automatically.	**boolean**	**true**
quarkus.camel.service.discovery.exclude-patterns A comma-separated list of Ant-path style patterns to match Camel service definition files in the classpath. The services defined in the matching files will not be discoverable via the **org.apache.camel.spi.FactoryFinder** mechanism. The excludes have higher precedence than includes. The excludes defined here can also be used to veto the discoverability of services included by Camel Quarkus extensions. Example values: **META-INF/services/org/apache/camel/foo/*,META-INF/services/org/apache/camel/foo/**/bar**	**string**	

Configuration property	Type	Default
quarkus.camel.service.discovery.include-patterns A comma-separated list of Ant-path style patterns to match Camel service definition files in the classpath. The services defined in the matching files will be discoverable via the **org.apache.camel.spi.FactoryFinder** mechanism unless the given file is excluded via **exclude-patterns**. Note that Camel Quarkus extensions may include some services by default. The services selected here added to those services and the exclusions defined in **exclude-patterns** are applied to the union set. Example values: **META-INF/services/org/apache/camel/foo/*,META-INF/services/org/apache/camel/foo/**/bar**	**string**	
quarkus.camel.service.registry.exclude-patterns A comma-separated list of Ant-path style patterns to match Camel service definition files in the classpath. The services defined in the matching files will not be added to Camel registry during application's static initialization. The excludes have higher precedence than includes. The excludes defined here can also be used to veto the registration of services included by Camel Quarkus extensions. Example values: **META-INF/services/org/apache/camel/foo/*,META-INF/services/org/apache/camel/foo/**/bar**	**string**	
quarkus.camel.service.registry.include-patterns A comma-separated list of Ant-path style patterns to match Camel service definition files in the classpath. The services defined in the matching files will be added to Camel registry during application's static initialization unless the given file is excluded via **exclude-patterns**. Note that Camel Quarkus extensions may include some services by default. The services selected here added to those services and the exclusions defined in **exclude-patterns** are applied to the union set. Example values: **META-INF/services/org/apache/camel/foo/*,META-INF/services/org/apache/camel/foo/**/bar**	**string**	

Configuration property	Type	Default
quarkus.camel.runtime-catalog.components If **true** the Runtime Camel Catalog embedded in the application will contain JSON schemas of Camel components available in the application; otherwise component JSON schemas will not be available in the Runtime Camel Catalog and any attempt to access those will result in a RuntimeException. Setting this to **false** helps to reduce the size of the native image. In JVM mode, there is no real benefit of setting this flag to **false** except for making the behavior consistent with native mode.	**boolean**	**true**
quarkus.camel.runtime-catalog.languages If **true** the Runtime Camel Catalog embedded in the application will contain JSON schemas of Camel languages available in the application; otherwise language JSON schemas will not be available in the Runtime Camel Catalog and any attempt to access those will result in a RuntimeException. Setting this to **false** helps to reduce the size of the native image. In JVM mode, there is no real benefit of setting this flag to **false** except for making the behavior consistent with native mode.	**boolean**	**true**
quarkus.camel.runtime-catalog.dataformats If **true** the Runtime Camel Catalog embedded in the application will contain JSON schemas of Camel data formats available in the application; otherwise data format JSON schemas will not be available in the Runtime Camel Catalog and any attempt to access those will result in a RuntimeException. Setting this to **false** helps to reduce the size of the native image. In JVM mode, there is no real benefit of setting this flag to **false** except for making the behavior consistent with native mode.	**boolean**	**true**
quarkus.camel.runtime-catalog-models If **true** the Runtime Camel Catalog embedded in the application will contain JSON schemas of Camel EIP models available in the application; otherwise EIP model JSON schemas will not be available in the Runtime Camel Catalog and any attempt to access those will result in a RuntimeException. Setting this to **false** helps to reduce the size of the native image. In JVM mode, there is no real benefit of setting this flag to **false** except for making the behavior consistent with native mode.	**boolean**	**true**
quarkus.camel.routes-discovery.enabled Enable automatic discovery of routes during static initialization.	**boolean**	**true**

Configuration property	Type	Default
quarkus.camel.routes-discovery.exclude-patterns Used for exclusive filtering scanning of RouteBuilder classes. The exclusive filtering takes precedence over inclusive filtering. The pattern is using Ant-path style pattern. Multiple patterns can be specified separated by comma. For example to exclude all classes starting with Bar use: **/Bar* To exclude all routes form a specific package use: com/mycompany/bar/* To exclude all routes form a specific package and its sub-packages use double wildcards: com/mycompany/bar/** And to exclude all routes from two specific packages use: com/mycompany/bar/*,com/mycompany/stuff/*	**string**	
quarkus.camel.routes-discovery.include-patterns Used for inclusive filtering scanning of RouteBuilder classes. The exclusive filtering takes precedence over inclusive filtering. The pattern is using Ant-path style pattern. Multiple patterns can be specified separated by comma. For example to include all classes starting with Foo use: **/Foo* To include all routes form a specific package use: com/mycompany/foo/* To include all routes form a specific package and its sub-packages use double wildcards: com/mycompany/foo/** And to include all routes from two specific packages use: com/mycompany/foo/*,com/mycompany/stuff/*	**string**	
quarkus.camel.native.resources.exclude-patterns A comma separated list of Ant-path style patterns to match resources that should be excluded from the native executable. By default, resources not selected by quarkus itself are ignored. Then, inclusion of additional resources could be triggered with **includePatterns**. When the inclusion patterns is too large, eviction of previously selected resources could be triggered with **excludePatterns**.	**string**	
quarkus.camel.native.resources.include-patterns A comma separated list of Ant-path style patterns to match resources that should be included in the native executable. By default, resources not selected by quarkus itself are ignored. Then, inclusion of additional resources could be triggered with **includePatterns**. When the inclusion patterns is too large, eviction of previously selected resources could be triggered with **excludePatterns**.	**string**	

Configuration property	Type	Default
quarkus.camel.native.reflection.exclude-patterns A comma separated list of Ant-path style patterns to match class names that should be excluded from registering for reflection. Use the class name format as returned by the **java.lang.Class.getName()** method: package segments delimited by period **.** and inner classes by dollar sign **$**. This option narrows down the set selected by **include-patterns**. By default, no classes are excluded. This option cannot be used to unregister classes which have been registered internally by Quarkus extensions.	**string**	
quarkus.camel.native.reflection.include-patterns A comma separated list of Ant-path style patterns to match class names that should be registered for reflection. Use the class name format as returned by the **java.lang.Class.getName()** method: package segments delimited by period **.** and inner classes by dollar sign **$**. By default, no classes are included. The set selected by this option can be narrowed down by **exclude-patterns**. Note that Quarkus extensions typically register the required classes for reflection by themselves. This option is useful in situations when the built in functionality is not sufficient. Note that this option enables the full reflective access for constructors, fields and methods. If you need a finer grained control, consider using **io.quarkus.runtime.annotations.RegisterForReflection** annotation in your Java code. For this option to work properly, the artifacts containing the selected classes must either contain a Jandex index (**META-INF/jandex.idx**) or they must be registered for indexing using the **quarkus.index-dependency.*** family of options in**application.properties** - e.g. quarkus.index-dependency.my-dep.group-id = org.my-group quarkus.index-dependency.my-dep.artifact-id = my-artifact where **my-dep** is a label of your choice to tell Quarkus that **org.my-group** and with **my-artifact** belong together.	**string**	
quarkus.camel.csimple.on-build-time-analysis-failure What to do if it is not possible to extract CSimple expressions from a route definition at build time.	**org.apache.camel.quarkus.core.CamelConfig.FailureRemedy**	**warn**
quarkus.camel.main.enabled If **true** all **camel-main** features are enabled; otherwise no **camel-main** features are enabled.	**boolean**	**true**

Configuration property	Type	Default
quarkus.camel.main.shutdown.timeout A timeout (with millisecond precision) to wait for **CamelMain#stop()** to finish	**java.time.Duration**	**PT3S**
quarkus.camel.main.arguments.on-unknown The action to take when **CamelMain** encounters an unknown argument. fail - Prints the **CamelMain** usage statement and throws a **RuntimeException** ignore - Suppresses any warnings and the application startup proceeds as normal warn - Prints the **CamelMain** usage statement but allows the application startup to proceed as normal	**org.apache.camel.quarkus.core.CamelConfig.FailureRemedy**	**warn**

Configuration property fixed at build time. All other configuration properties are overridable at runtime.

2.11. DIRECT

Call another endpoint from the same Camel Context synchronously.

2.11.1. What's inside

- Direct component, URI syntax: **direct:name**

Please refer to the above link for usage and configuration details.

2.11.2. Maven coordinates

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

2.12. ELASTICSEARCH REST

Send requests to with an ElasticSearch via REST API.

2.12.1. What's inside

- Elasticsearch Rest component, URI syntax: **elasticsearch-rest:clusterName**

Please refer to the above link for usage and configuration details.

2.12.2. Maven coordinates

```
<dependency>
```

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

2.13. FILE

Read and write files.

2.13.1. What's inside

- File component, URI syntax: **file:directoryName**

Please refer to the above link for usage and configuration details.

2.13.2. Maven coordinates

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

2.14. FTP

Upload and download files to/from FTP or SFTP servers.

2.14.1. What's inside

- FTP component, URI syntax: **ftp:host:port/directoryName**
- FTPS component, URI syntax: **ftps:host:port/directoryName**
- SFTP component, URI syntax: **sftp:host:port/directoryName**

Please refer to the above links for usage and configuration details.

2.14.2. Maven coordinates

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

2.15. HL7

Marshal and unmarshal HL7 (Health Care) model objects using the HL7 MLLP codec.

2.15.1. What's inside

- HL7 data format
- HL7 Terser language

Please refer to the above links for usage and configuration details.

2.15.2. Maven coordinates

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

2.15.3. Camel Quarkus limitations

For MLLP with TCP, Netty is the only supported means of running an Hl7 MLLP listener. Mina is not supported since it has no GraalVM native support at present.

Optional support for **HL7MLLPNettyEncoderFactory** & **HL7MLLPNettyDecoderFactory** codecs can be obtained by adding a dependency in your project **pom.xml** to **camel-quarkus-netty**.

2.16. HTTP

Send requests to external HTTP servers using Apache HTTP Client 4.x.

2.16.1. What's inside

- HTTP component, URI syntax: **http://httpUri**
- HTTPS (Secure) component, URI syntax: **https://httpUri**

Please refer to the above links for usage and configuration details.

2.16.2. Maven coordinates

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

2.16.3. SSL in native mode

This extension auto-enables SSL support in native mode. Hence you do not need to add **quarkus.ssl.native=true** to your **application.properties** yourself. See also Quarkus SSL guide.

2.17. JACKSON

Marshal POJOs to JSON and back using Jackson

2.17.1. What's inside

- JSON Jackson data format

Please refer to the above link for usage and configuration details.

2.17.2. Maven coordinates

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

2.18. AVRO JACKSON

Marshal POJOs to Avro and back using Jackson.

2.18.1. What's inside

- Avro Jackson data format

Please refer to the above link for usage and configuration details.

2.18.2. Maven coordinates

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

2.19. PROTOBUF JACKSON

Marshal POJOs to Protobuf and back using Jackson.

2.19.1. What's inside

- Protobuf Jackson data format

Please refer to the above link for usage and configuration details.

2.19.2. Maven coordinates

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

2.20. JACKSONXML

Unmarshal a XML payloads to POJOs and back using XMLMapper extension of Jackson.

2.20.1. What's inside

- JacksonXML data format

Please refer to the above link for usage and configuration details.

2.20.2. Maven coordinates

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

2.21. JIRA

Interact with JIRA issue tracker.

2.21.1. What's inside

- Jira component, URI syntax: **jira:type**

Please refer to the above link for usage and configuration details.

2.21.2. Maven coordinates

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

2.21.3. SSL in native mode

This extension auto-enables SSL support in native mode. Hence you do not need to add **quarkus.ssl.native=true** to your **application.properties** yourself. See also Quarkus SSL guide.

2.22. JMS

Sent and receive messages to/from a JMS Queue or Topic.

2.22.1. What's inside

- JMS component, URI syntax: **jms:destinationType:destinationName**

Please refer to the above link for usage and configuration details.

2.22.2. Maven coordinates

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

2.22.3. Usage

2.22.3.1. Message mapping with **org.w3c.dom.Node**

The Camel JMS component supports message mapping between **javax.jms.Message** and **org.apache.camel.Message**. When wanting to convert a Camel message body type of **org.w3c.dom.Node**, you must ensure that the **camel-quarkus-jaxp** extension is present on the classpath.

2.23. JSON PATH

Evaluate a JsonPath expression against a JSON message body.

2.23.1. What's inside

- JsonPath language

Please refer to the above link for usage and configuration details.

2.23.2. Maven coordinates

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

2.24. JTA

Enclose Camel routes in the transactions using Java Transaction API (JTA) and Narayana transaction manager

2.24.1. What's inside

- JTA

Please refer to the above link for usage and configuration details.

2.24.2. Maven coordinates

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

2.24.3. Usage

This extension should be added when you need to use the **transacted()** EIP in the router. It leverages the transaction capabilities provided by the narayana-jta extension in Quarkus.

Refer to the Quarkus Transaction guide for the more details about transaction support. For a simple usage:

```
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")
```

Support is provided for various transaction policies.

Policy	Description
PROPAGATION_MANDATORY	Support a current transaction; throw an exception if no current transaction exists.
PROPAGATION_NEVER	Do not support a current transaction; throw an exception if a current transaction exists.
PROPAGATION_NOT_SUPPORTED	Do not support a current transaction; rather always execute non-transactionally.
PROPAGATION_REQUIRED	Support a current transaction; create a new one if none exists.
PROPAGATION_REQUIRES_NEW	Create a new transaction, suspending the current transaction if one exists.
PROPAGATION_SUPPORTS	Support a current transaction; execute non-transactionally if none exists.

2.25. KAFKA

Sent and receive messages to/from an Apache Kafka broker.

2.25.1. What's inside

- Kafka component, URI syntax: **kafka:topic**

Please refer to the above link for usage and configuration details.

2.25.2. Maven coordinates

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

2.26. KAMELET

The Kamelet Component provides support for interacting with the Camel Route Template engine

2.26.1. What's inside

- Kamelet component, URI syntax: **kamelet:templateId/routeId**

Please refer to the above link for usage and configuration details.

2.26.2. Maven coordinates

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

NOTE

Automatic discovery of kamelet definitions having the extension **.kamelet.yaml** is provided in the community version of **camel-kamelet**. Please note that this automatic discovery is not supported by Red Hat in the **camel-quarkus-kamelet** extension for TP2.

2.27. LOG

Log messages to the underlying logging mechanism.

2.27.1. What's inside

- Log component, URI syntax: **log:loggerName**

Please refer to the above link for usage and configuration details.

2.27.2. Maven coordinates

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

2.28. MAIN

Bootstrap Camel using Camel Main which brings advanced auto-configuration capabilities and integration with Quarkus Command Mode

2.28.1. Maven coordinates

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

2.29. MICROPROFILE HEALTH

Bridging Eclipse MicroProfile Health with Camel health checks

2.29.1. What's inside

- Microprofile Health

Please refer to the above link for usage and configuration details.

2.29.2. Maven coordinates

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

2.29.3. Usage

By default, classes extending **AbstractHealthCheck** are registered as both liveness and readiness checks. You can override the **isReadiness** method to control this behaviour.

Any checks provided by your application are automatically discovered and bound to the Camel registry. They will be available via the Quarkus health endpoints **/health/live** and **/health/ready**.

You can also provide custom **HealthCheckRepository** implementations and these are also automatically discovered and bound to the Camel registry for you.

Refer to the Quarkus health guide for further information.

2.29.3.1. Provided health checks

Some checks are automatically registered for your application.

2.29.3.1.1. Camel Context Health

Inspects the Camel Context status and causes the health check status to be **DOWN** if the status is anything other than 'Started'.

2.29.3.1.2. Camel Route Health

Inspects the status of each route and causes the health check status to be **DOWN** if any route status is not 'Started'.

2.29.4. Additional Camel Quarkus configuration

Configuration property	Type	Default
quarkus.camel.health.enabled Set whether to enable Camel health checks	**boolean**	**true**

Configuration property fixed at build time. All other configuration properties are overridable at runtime.

2.30. MICROPROFILE METRICS

Expose metrics from Camel routes.

2.30.1. What's inside

- MicroProfile Metrics component, URI syntax: **microprofile-metrics:metricType:metricName**

Please refer to the above link for usage and configuration details.

2.30.2. Maven coordinates

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

2.30.3. Usage

The microprofile-metrics component automatically exposes a set of Camel application metrics. Some of these include:

2.30.3.1. Camel Context metrics

Metric Name	Type
camel.context.status The status of the Camel Context represented by the **ServiceStatus** enum ordinal	Gauge
camel.context.uptime The Camel Context uptime in milliseconds	Gauge
camel.context.exchanges.completed.total The total number of completed exchanges	Counter
camel.context.exchanges.failed.total The total number of failed exchanges	Counter
camel.context.exchanges.inflight.total The total number of inflight exchanges	Gauge
camel.context.exchanges.total The total number of all exchanges	Counter

Metric Name	Type
camel.context.externalRedeliveries.total The total number of all external redeliveries	Counter
camel.context.failuresHandled.total The total number of all failures handled	Counter

2.30.3.2. Camel Route metrics

Metric Name	Type
camel.route.count The number of routes	Gauge
camel.route.running.count The number of running routes	Gauge
camel.route.exchanges.completed.total The total number of completed exchanges for the route	Counter
camel.route.exchanges.failed.total The total number of failed exchanges for the route	Counter
camel.route.exchanges.inflight.total The total number of inflight exchanges for the route	Gauge
camel.route.exchanges.total The total number of all exchanges for the route	Counter
camel.route.externalRedeliveries.total The total number of all external redeliveries for the route	Counter
camel.route.failuresHandled.total The total number of all failures handled for the route	Counter

All metrics are tagged with the name of the Camel Context and the id of the route where applicable.

You can also produce your own customized metrics in your Camel routes. For more information, refer to the microprofile-metrics component documentation.

Metrics are exposed to Quarkus as application metrics and they can be browsed at http://localhost:8080/metrics/application.

2.30.4. Additional Camel Quarkus configuration

Configuration property	Type	Default
quarkus.camel.metrics.enable-route-policy Set whether to enable the MicroProfileMetricsRoutePolicyFactory for capturing metrics on route processing times.	**boolean**	**true**
quarkus.camel.metrics.enable-message-history Set whether to enable the MicroProfileMetricsMessageHistoryFactory for capturing metrics on individual route node processing times. Depending on the number of configured route nodes, there is the potential to create a large volume of metrics. Therefore, this option is disabled by default.	**boolean**	**false**
quarkus.camel.metrics.enable-exchange-event-notifier Set whether to enable the MicroProfileMetricsExchangeEventNotifier for capturing metrics on exchange processing times.	**boolean**	**true**
quarkus.camel.metrics.enable-route-event-notifier Set whether to enable the MicroProfileMetricsRouteEventNotifier for capturing metrics on the total number of routes and total number of routes running.	**boolean**	**true**
quarkus.camel.metrics.enable-camel-context-event-notifier Set whether to enable the MicroProfileMetricsCamelContextEventNotifier for capturing metrics about the CamelContext, such as status and uptime.	**boolean**	**true**

Configuration property fixed at build time. All other configuration properties are overridable at runtime.

2.31. MLLP

Communicate with external systems using the MLLP protocol.

2.31.1. What's inside

- MLLP component, URI syntax: **mllp:hostname:port**

Please refer to the above link for usage and configuration details.

2.31.2. Maven coordinates

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

2.32. MOCK

Test routes and mediation rules using mocks.

2.32.1. What's inside

- Mock component, URI syntax: **mock:name**

Please refer to the above link for usage and configuration details.

2.32.2. Maven coordinates

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

2.32.3. Usage

To use camel-mock capabilities in tests it is required to get access to MockEndpoint instances.

CDI injection could be used for accessing instances (see Quarkus documentation). You can inject camelContext into test using **@Inject** annotation. Camel context can be then used for obtaining mock endpoints. See the following example:

```
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();
    }
}
```

Route used for the example test:

```
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");
    }
}
```

2.32.4. Camel Quarkus limitations

Injection of CDI beans (described in Usage) does not work in native mode.

In the native mode the test and the application under test are running in two different processes and it is not possible to share a mock bean between them (see Quarkus documentation).

2.33. MONGODB

Perform operations on MongoDB documents and collections.

2.33.1. What's inside

- MongoDB component, URI syntax: **mongodb:connectionBean**

Please refer to the above link for usage and configuration details.

2.33.2. Maven coordinates

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

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

2.33.3. Additional Camel Quarkus configuration

The extension leverages the Quarkus MongoDB Client extension. The Mongo client can be configured via the Quarkus MongoDB Client configuration options.

The Camel Quarkus MongoDB extension automatically registers a MongoDB client bean named **camelMongoClient**. This can be referenced in the mongodb endpoint URI **connectionBean** path parameter. For example:

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

If your application needs to work with multiple MongoDB servers, you can create a "named" client and reference in your route by injecting a client and the related configuration as explained in the Quarkus MongoDB extension client injection. For example:

```
//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");
        }
    }
```

Note that when using named clients, the "default" **camelMongoClient** bean will still be produced. Refer to the Quarkus documentation on Multiple MongoDB Clients for more information.

2.34. NETTY

Socket level networking using TCP or UDP with the Netty 4.x.

2.34.1. What's inside

- Netty component, URI syntax: **netty:protocol://host:port**

Please refer to the above link for usage and configuration details.

2.34.2. Maven coordinates

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

2.35. PLATFORM HTTP

This extension allows for creating HTTP endpoints for consuming HTTP requests.

It is built on top of Eclipse Vert.x Web service provided by the **quarkus-vertx-web** extension.

2.35.1. What's inside

- Platform HTTP component, URI syntax: **platform-http:path**

Please refer to the above link for usage and configuration details.

2.35.2. Maven coordinates

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

2.35.3. Usage

2.35.3.1. Basic Usage

Serve all HTTP methods on the /**hello** endpoint:

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

Serve only GET requests on the /**hello** endpoint:

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

2.35.3.2. Using platform-http via Camel REST DSL

To be able to use Camel REST DSL with the **platform-http** component, add **camel-quarkus-rest** in addition to **camel-quarkus-platform-http** to your **pom.xml**:

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

Then you can use the Camel REST DSL:

```
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. Handling multipart/form-data file uploads

If you want Camel Quarkus to attach uploaded files to Camel messages for you, you need to add the following optional dependency to your **pom.xml**:

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

You can restrict the uploads to certain file extensions by white listing them:

```
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);
                }
            });
        }
    });
```

Also check the **quarkus.http.body.*** configuration options in Quarkus documentation, esp. **quarkus.http.body.handle-file-uploads**, **quarkus.http.body.uploads-directory** and **quarkus.http.body.delete-uploaded-files-on-end**.

2.36. REST

Expose REST services and their OpenAPI Specification or call external REST services.

2.36.1. What's inside

- REST component, URI syntax: **rest:method:path:uriTemplate**
- REST API component, URI syntax: **rest-api:path/contextIdPattern**

Please refer to the above links for usage and configuration details.

2.36.2. Maven coordinates

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

2.36.3. Additional Camel Quarkus configuration

This extension depends on the Platform HTTP extension and configures it as the component that provides the REST transport.

To use another REST transport provider, such as **netty-http** or **servlet**, you need to add the respective extension as a dependency to your project and set the provider in your **RouteBuilder**. E.g. for **servlet**, you'd have to add the **org.apache.camel.quarkus:camel-quarkus-servlet** dependency and the set the provider as follows:

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

public class CamelRoute extends RouteBuilder {

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

2.37. SALESFORCE

Communicate with Salesforce using Java DTOs.

2.37.1. What's inside

- Salesforce component, URI syntax: **salesforce:operationName:topicName**

Please refer to the above link for usage and configuration details.

2.37.2. Maven coordinates

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

2.37.3. SSL in native mode

This extension auto-enables SSL support in native mode. Hence you do not need to add **quarkus.ssl.native=true** to your **application.properties** yourself. See also Quarkus SSL guide.

2.38. XQUERY

Query and/or transform XML payloads using XQuery and Saxon.

2.38.1. What's inside

- XQuery component, URI syntax: **xquery:resourceUri**
- XQuery language

Please refer to the above links for usage and configuration details.

2.38.2. Maven coordinates

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

2.38.3. Additional Camel Quarkus configuration

Beyond standard usages described above, a trick is needed when using the xquery language and component with classpath resources in native mode. In such a situation, one needs to explicitly embed the resources in the native executable by specifying the **include-patterns** option.

For instance, the 2 routes below would load a xquery script from 2 classpath resources respectively named *myxquery.txt* and *another-xquery.txt*:

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

In order to work in native mode the **include-patterns** configuration should be set. For instance, in the **application.properties** file as below :

```
quarkus.camel.native.resources.include-patterns = *.txt
```

2.39. SEDA

Asynchronously call another endpoint from any Camel Context in the same JVM.

2.39.1. What's inside

- SEDA component, URI syntax: **seda:name**

Please refer to the above link for usage and configuration details.

2.39.2. Maven coordinates

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

-

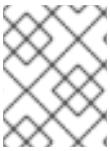
2.40. SOAP DATAFORMAT

Marshal Java objects to SOAP messages and back.

2.40.1. What's inside

- SOAP data format

Please refer to the above link for usage and configuration details.



NOTE

The proxy example referred to in the SOAP data format documentation is not supported in this TP2 release.

2.40.2. Maven coordinates

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

2.41. SQL

Perform SQL queries using Spring JDBC.

2.41.1. What's inside

- SQL component, URI syntax: **sql:query**
- SQL Stored Procedure component, URI syntax: **sql-stored:template**

Please refer to the above links for usage and configuration details.

2.41.2. Maven coordinates

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

2.41.3. Additional Camel Quarkus configuration

When configuring **sql** or **sql-stored** endpoints to reference script files from the classpath, set the following configuration property to ensure that they are available in native mode.

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

2.42. TIMER

Generate messages in specified intervals using java.util.Timer.

2.42.1. What's inside

- Timer component, URI syntax: **timer:timerName**

Please refer to the above link for usage and configuration details.

2.42.2. Maven coordinates

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

2.43. XPATH

Evaluate an XPath expression against an XML payload.

2.43.1. What's inside

- XPath language

Please refer to the above link for usage and configuration details.

2.43.2. Maven coordinates

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

2.43.3. Additional Camel Quarkus configuration

Beyond standard usages, a trick is needed when using the xpath language with classpath resources in native mode. In such a situation, one needs to explicitly embed the resources in the native executable by specifying the **include-patterns** option.

For instance, the route below would load an xpath script from a classpath resource named *myxpath.txt*:

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

In order to work in native mode the **include-patterns** configuration should be set. For instance, in the **application.properties** file as below :

```
quarkus.camel.native.resources.include-patterns = *.txt
```