Red Hat build of Apache Camel 4.0

Migrating Fuse 7 Applications to Red Hat build of Apache Camel for Quarkus
Red Hat build of Apache Camel 4.0 Migrating Fuse 7 Applications to Red Hat build of Apache Camel for Quarkus

Migrating Fuse 7 Applications to Red Hat build of Apache Camel for Quarkus
Abstract

Migrating Fuse 7 Applications to Red Hat build of Apache Camel for Quarkus provides information on migrating from Red Hat Fuse 7 to Red Hat build of Apache Camel for Quarkus.
**Table of Contents**

PREFACE .................................................................................................................. 4
   MAKING OPEN SOURCE MORE INCLUSIVE ......................................................... 4

CHAPTER 1. OVERVIEW OF MIGRATING FUSE 7 APPLICATIONS TO RED HAT BUILD OF APACHE CAMEL FOR QUARKUS .................................................................................. 5
   1. STANDARD MIGRATION PATHS ..................................................................... 5
      1.1. XML path ................................................................................................. 5
      1.2. Java path .................................................................................................. 6
   2. ARCHITECTURAL CHANGES ...................................................................... 6

CHAPTER 2. MIGRATING CAMEL ROUTES FROM FUSE 7 TO RED HAT BUILD OF APACHE CAMEL FOR QUARKUS ................................................................. 7
   1. JAVA DSL ROUTE MIGRATION EXAMPLE ................................................... 7
   2. BLUEPRINT XML DSL ROUTE MIGRATION ............................................... 8
      2.2.1. XML-IO-DSL limitations ................................................................... 8
      2.2.2. Blueprint XML DSL route migration example .................................. 10

CHAPTER 3. MIGRATING APACHE CAMEL ......................................................... 13
   1. ABOUT THE CAMEL MIGRATION GUIDE ..................................................... 13
   2. MIGRATING TO APACHE CAMEL 4 ................................................................. 13
      2.1. Java versions ............................................................................................ 13
      2.2. Removed Components ............................................................................ 13
      2.3. Logging .................................................................................................... 15
      2.4. JUnit 4 ..................................................................................................... 15
      2.5. API Changes ............................................................................................ 15
      2.6. EIP Changes ............................................................................................ 16
         2.6.1. Poll Enrich EIP .................................................................................. 16
         2.6.2. CircuitBreaker EIP ........................................................................... 17
      2.7. XML DSL ................................................................................................ 17
      2.8. Type Converter ....................................................................................... 17
      2.9. Tracing ..................................................................................................... 18
      2.10. UseOriginalMessage / UseOriginalBody ............................................. 18
      2.11. Camel Health ......................................................................................... 18
         2.11.1. Producer Health Checks ................................................................. 18
      2.12. JMX ........................................................................................................ 18
      2.13. YAML DSL ............................................................................................. 18
      2.15. XML serialization ............................................................................... 19
      2.16. OpenAPI Maven Plugin ....................................................................... 19
      2.17. Component changes ............................................................................ 19
         2.17.1. Category ......................................................................................... 19
         2.17.2. camel-openapi-rest-dsl-generator .............................................. 19
         2.17.3. camel-atom .................................................................................... 19
         2.17.4. camel-azure-cosmosdb ................................................................. 20
         2.17.5. camel-bean .................................................................................... 20
         2.17.6. camel-box ..................................................................................... 20
         2.17.7. camel-caffeine ............................................................................... 20
         2.17.8. camel-fhir ...................................................................................... 20
         2.17.9. camel-google ............................................................................... 20
         2.17.10. camel-http .................................................................................. 20
         2.17.11. camel-http-common ................................................................. 21
         2.17.12. camel-kubernetes ...................................................................... 21
Red Hat build of Apache Camel 4.0 Migrating Fuse 7 Applications to Red Hat build of Apache Camel for Quarkus

3.2.17.13. camel-web3j
3.2.17.14. camel-main
3.2.17.15. camel-micrometer
3.2.17.16. camel-jbang
3.2.17.17. camel-openapi-java
3.2.17.18. camel-optaplanner
3.2.17.19. camel-platform-http-vertx
3.2.17.20. camel-salesforce
3.2.17.21. camel-slack
3.2.17.22. camel-micrometer-starter
3.2.17.23. camel-platform-http-starter
3.2.17.24. camel-twitter

3.3. MIGRATING TO APACHE CAMEL 3
3.3.1. Java versions
3.3.1.1. JAXB removed in JDK 11
3.3.2. Modularization of camel-core
3.3.3. Modularization of Components
3.3.4. Multiple CamelContexts per application not supported
3.3.5. Deprecated APIs and Components
3.3.5.1. Removed components
3.3.5.2. Renamed components
3.3.6. Changes to Camel components
3.3.6.1. Mock component
3.3.6.2. ActiveMQ
3.3.6.3. AWS
3.3.6.4. Camel CXF
3.3.6.4.1. Camel CXF changed namespaces
3.3.6.5. FHIR
3.3.6.6. Kafka
3.3.6.7. Telegram
3.3.6.8. JMX
3.3.6.9. XSLT
3.3.6.10. XML DSL Migration
3.3.7. Migrating Camel Maven Plugins

CHAPTER 4. MIGRATING CAMEL QUARKUS PROJECTS .................................................. 31
4.1. UPDATING PROJECTS TO THE LATEST QUARKUS VERSION 31
4.1.1. Prerequisites 31
4.1.2. Updating with Maven 31

CHAPTER 5. ADDITIONAL RESOURCES ................................................................. 33
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. OVERVIEW OF MIGRATING FUSE 7 APPLICATIONS TO RED HAT BUILD OF APACHE CAMEL FOR QUARKUS

Fuse

Red Hat Fuse is an agile integration solution based on open source communities like Apache Camel and Apache Karaf. Red Hat Fuse is a lightweight, flexible integration platform that enables rapid on-premise cloud integration.

You can run Red Hat Fuse using three different runtimes:

- Karaf which supports OSGi applications
- Spring Boot
- JBoss EAP (Enterprise Application Platform)

Red Hat build of Apache Camel for Quarkus

Red Hat build of Apache Camel for Quarkus brings the integration capabilities of Apache Camel and its vast component library to the Quarkus runtime. Red Hat build of Camel Quarkus provides Quarkus extensions for many of the Camel components.

Camel Quarkus takes advantage of the many performance improvements made in Camel 3, which results in a lower memory footprint, less reliance on reflection, and faster startup times.

In a Red Hat build of Apache Camel for Quarkus application, you define Camel routes using Java DSL, so you can migrate the Camel routes that you use in your Fuse application to CEQ.

Camel on EAP

Karaf, which follows the OSGi dependency management concept, and EAP, which follows the JEE specification, are application servers impacted by the adoption of containerized applications.

Containers have emerged as the predominant method for packaging applications. Consequently, the responsibility for managing applications, which encompasses deployment, scaling, clustering, and load balancing, has shifted from the application server to the container orchestration using Kubernetes.

Although EAP continues to be supported on Red Hat Openshift, Camel 3 is no longer supported on an EAP server. So if you have a Fuse 7 application running on an EAP server, you should consider migrating your application to the Red Hat Build of Apache Camel for Spring Boot or the Red Hat build of Apache Camel for Quarkus and take the benefit of the migration process to consider a redesign, or partial redesign of your application, from a monolith to a microservices architecture.

If you do not use Openshift, RHEL virtual machines remain a valid approach when you deploy your application for Spring Boot and Quarkus, and Quarkus also benefits from its native compilation capabilities. It is important to evaluate the tooling to support the management of a microservices architecture on such a platform.

Red Hat provides this capability through Ansible, using the Red Hat Ansible for Middleware collections.

1.1. STANDARD MIGRATION PATHS

1.1.1. XML path
Fuse applications written in Spring XML or Blueprint XML should be migrated towards an XML-based flavor, and can target either the Spring Boot or the Quarkus runtime with no difference in the migration steps.

1.1.2. Java path

Fuse applications written in Java DSL should be migrated towards a Java-based flavor, and can target either the Spring Boot or the Quarkus runtime with no difference in the migration steps.

1.2. ARCHITECTURAL CHANGES

Openshift has replaced Fabric8 as the runtime platform for Fuse 6 users and is the recommended target for your Fuse application migration.

You should consider the following architectural changes when you are migrating your application:

- If your Fuse 6 application relied on the Fabric8 service discovery, you should use Kubernetes Service Discovery when running Camel 3 on OpenShift.

- If your Fuse 6 application relies on OSGi bundle configuration, you should use Kubernetes ConfigMaps and Secrets when running Camel 3 on OpenShift.

- If your application uses a file-based route definition, consider using AWS S3 technology when running Camel 3 on OpenShift.

- If your application uses a standard filesystem, the resulting Spring Boot or Quarkus applications should be deployed on standard RHEL virtual machines rather than the OpenShift platform.

- Delegation of Inbound HTTPS connections to the OpenShift Router which handles SSL requirements.

- Delegation of Hystrix features to Service Mesh.
NOTE
You can define Camel routes in Red Hat build of Apache Camel for Quarkus applications using Java DSL, XML IO DSL, or YAML.

2.1. JAVA DSL ROUTE MIGRATION EXAMPLE

To migrate a Java DSL route definition from your Fuse application to CEQ, you can copy your existing route definition directly to your Red Hat build of Apache Camel for Quarkus application and add the necessary dependencies to your Red Hat build of Apache Camel for Quarkus pom.xml file.

In this example, we will migrate a content-based route definition from a Fuse 7 application to a new CEQ application by copying the Java DSL route to a file named `Routes.java` in your CEQ application.

Procedure

1. Using the `code.quarkus.redhat.com` website, select the extensions required for this example:
   - camel-quarkus-file
   - camel-quarkus-xpath

2. Navigate to the directory where you extracted the generated project files from the previous step:
   ```bash
   $ cd <directory_name>
   ```


4. Add the route definition from your Fuse application to the `Routes.java`, similar to the following example:

   ```java
   package org.acme;

   import org.apache.camel.builder.RouteBuilder;

   public class Routes extends RouteBuilder {
   // Add your Java DSL route definition here
   
   public void configure() {
       from("file:work/cbr/input")
       .log("Receiving order ${file:name}")
       .choice()
       .when().xpath("//order/customer/country[text() = 'UK']")
       .log("Sending order ${file:name} to the UK")
       .to("file:work/cbr/output/uk")
       .when().xpath("//order/customer/country[text() = 'US']")
       .log("Sending order ${file:name} to the US")
       .to("file:work/cbr/output/uk")
       .otherwise()
       .log("Sending order ${file:name} to another country")
       .to("file:work/cbr/output/others");
   }
   ```
5. Compile your CEQ application.

mvn clean compile quarkus:dev

NOTE

This command compiles the project, starts your application, and lets the Quarkus tooling watch for changes in your workspace. Any modifications in your project will automatically take effect in the running application.

2.2. BLUEPRINT XML DSL ROUTE MIGRATION

To migrate a Blueprint XML route definition from your Fuse application to CEQ, use the camel-quarkus-xml-io-dsl extension and copy your Fuse application route definition directly to your Red Hat build of Apache Camel for Quarkus application. You will then need to add the necessary dependencies to the Red Hat build of Apache Camel for Quarkus pom.xml file and update your Red Hat build of Apache Camel for Quarkus configuration in the application.properties file.

NOTE

Red Hat build of Apache Camel for Quarkus supports Camel version 4, but Fuse 7 supports Camel version 2.

For more information relating to upgrading Camel when you migrate your Red Hat Fuse 7 application to CEQ, see:

- Migrating to Apache Camel 3
- Migrating to Apache Camel 4

For more information about using beans in Camel Quarkus, see the CDI and the Camel Bean Component section in the Developing Applications with Red Hat build of Apache Camel for Quarkus guide.

2.2.1. XML-IO-DSL limitations

You can use the camel-quarkus-xml-io-dsl extension to assist with migrating a Blueprint XML route definition to CEQ.

The camel-quarkus-xml-io-dsl extension only supports the following &lt;camelContext&gt; sub-elements:

- routeTemplates
- templatedRoutes
- rests
- routes
- routeConfigurations
NOTE

As Blueprint XML supports other bean definitions that are not supported by the camel-quarkus-xml-io-dsl extension, you may need to rewrite other bean definitions that are included in your Blueprint XML route definition.

You must define every element (XML IO DSL) in a separate file. For example, this is a simplified example of a Blueprint XML route definition:

```xml
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <camelContext xmlns="http://camel.apache.org/schema/blueprint">
    <restConfiguration contextPath="/camel"/>
    <rest path="/books">
      <get path="/">
        <to ..../>
      </get>
    </rest>
    <route>
      <from ..../>
    </route>
  </camelContext>
</blueprint>
```

You can migrate this Blueprint XML route definition to CEQ using XML IO DSL as defined in the following files:

`src/main/resources/routes/camel-rests.xml`

```xml
<rests xmlns="http://camel.apache.org/schema/spring">
  <rest path="/books">
    <get path="/">
      <to ..../>
    </get>
  </rest>
</rests>
```

`src/main/resources/routes/camel-routes.xml`

```xml
<routes xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from ..../>
  </route>
</routes>
```

You must use Java DSL to define other elements which are not supported, such as `<restConfiguration>`. For example, using a route builder defined in a `camel-rests.xml` file as follows:

`src/main/resources/routes/camel-rests.xml`

```java
import org.apache.camel.builder.RouteBuilder;
public class Routes extends RouteBuilder {
  public void configure() {
    restConfiguration()
```
2.2.2. Blueprint XML DSL route migration example

NOTE

For more information about using the XML IO DSL extension, see the XML IO DSL documentation in the Red Hat build of Apache Camel for Quarkus Reference.

In this example, you are migrating a content-based route definition from a Fuse application to a new CEQ application by copying the Blueprint XML route definition to a file named camel-routes.xml in your CEQ application.

Procedure

1. Using the code.quarkus.redhat.com website, select the following extensions for this example:
   - camel-quarkus-xml-io-dsl
   - camel-quarkus-file
   - camel-quarkus-xpath

2. Select Generate your application to confirm your choices and display the overlay screen with the download link for the archive that contains your generated project.

3. Select Download the ZIP to save the archive with the generated project files to your machine.

4. Extract the contents of the archive.

5. Navigate to the directory where you extracted the generated project files from the previous step:

   * cd <directory_name>


7. Copy the <route> element and sub-elements from the following blueprint-example.xml example to the camel-routes.xml file:

   ```xml
   <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
     <camelContext id="cbr-example-context" xmlns="http://camel.apache.org/schema/blueprint">
       <route id="cbr-route">
         <from id="_from1" url="file:work/cbr/input"/>
         <log id="_log1" message="Receiving order ${file:name}"/>
         <choice id="_choice1">
           <when id="_when1">
             <xpath id="_xpath1" order/customer/country = 'UK'</xpath>
             <log id="_log2" message="Sending order ${file:name} to the UK"/>
           </when id="_when1">
         </choice id="_choice1">
       </route id="cbr-route">
     </camelContext id="cbr-example-context" xmlns="http://camel.apache.org/schema/blueprint">
   </blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
8. Modify **application.properties**

```properties
# Camel
#
camel.context.name = camel-quarkus-xml-dsl-example
camel.main.routes-include-pattern = file:src/main/resources/routes/camel-routes.xml
```

9. Compile your CEQ application.

```
mvn clean compile quarkus:dev
```
NOTE

This command compiles the project, starts your application, and lets the Quarkus tooling watch for changes in your workspace. Any modifications in your project will automatically take effect in the running application.
CHAPTER 3. MIGRATING APACHE CAMEL

3.1. ABOUT THE CAMEL MIGRATION GUIDE

This guide details the changes in the Apache Camel components that you must consider when migrating your application. This guide provides information about following changes.

- Supported Java versions
- Changes to Apache Camel components and deprecated components
- Changes to APIs and deprecated APIs
- Updates to EIP
- Updated to tracing and health checks

3.2. MIGRATING TO APACHE CAMEL 4

This section provides information that can help you migrate your Apache Camel applications from version 3.20 or higher to 4.0.

**NOTE**

For information about individual versions, see:

- Apache Camel 3.x upgrade guide.
- Apache Camel 4.x upgrade guide.

For information about how to upgrade Apache Camel Quarkus, see:

- Camel Quarkus 3.2.0 migration guide.

3.2.1. Java versions

Apache Camel 4 supports Java 17. Support for Java 11 is dropped.

3.2.2. Removed Components

The following components have been removed:

<table>
<thead>
<tr>
<th>Component</th>
<th>Alternative component(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>camel-any23</td>
<td>none</td>
</tr>
<tr>
<td>camel-atlastmap</td>
<td>none</td>
</tr>
<tr>
<td>camel-atmos</td>
<td>none</td>
</tr>
<tr>
<td>camel-caffeine-lrucache</td>
<td>camel-cache, camel-ignite, camel-infinispan</td>
</tr>
<tr>
<td>Component</td>
<td>Alternative component(s)</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>camel-cdi</td>
<td>camel-spring-boot, camel-quarkus</td>
</tr>
<tr>
<td>camel-corda</td>
<td>none</td>
</tr>
<tr>
<td>camel-directvm</td>
<td>camel-direct</td>
</tr>
<tr>
<td>camel-dozer</td>
<td>camel-mapstruct</td>
</tr>
<tr>
<td>camel-elasticsearch-rest</td>
<td>camel-elasticsearch</td>
</tr>
<tr>
<td>camel-gora</td>
<td>none</td>
</tr>
<tr>
<td>camel-hbase</td>
<td>none</td>
</tr>
<tr>
<td>camel-hbase</td>
<td>none</td>
</tr>
<tr>
<td>camel-hbase</td>
<td>none</td>
</tr>
<tr>
<td>camel-hyperledger-aries</td>
<td>none</td>
</tr>
<tr>
<td>camel-iota</td>
<td>none</td>
</tr>
<tr>
<td>camel-ipfs</td>
<td>none</td>
</tr>
<tr>
<td>camel-jbpm</td>
<td>none</td>
</tr>
<tr>
<td>camel-jclouds</td>
<td>none</td>
</tr>
<tr>
<td>camel-johnzon</td>
<td>camel-jackson, camel-fastjson, camel-gson</td>
</tr>
<tr>
<td>camel-microprofile-metrics</td>
<td>camel-micrometer, camel-opentelemetry</td>
</tr>
<tr>
<td>camel-milo</td>
<td>none</td>
</tr>
<tr>
<td>camel-opentracing</td>
<td>camel-micrometer, camel-opentelemetry</td>
</tr>
<tr>
<td>camel-rabbitmq</td>
<td>spring-rabbitmq-component</td>
</tr>
<tr>
<td>camel-rest-swagger</td>
<td>camel-openapi-rest</td>
</tr>
<tr>
<td>camel-restdsl-swagger-plugin</td>
<td>camel-restdsl-openapi-plugin</td>
</tr>
<tr>
<td>camel-resteasy</td>
<td>camel-cxf, camel-rest</td>
</tr>
<tr>
<td>camel-solr</td>
<td>none</td>
</tr>
<tr>
<td>camel-spark</td>
<td>none</td>
</tr>
<tr>
<td>camel-spring-integration</td>
<td>none</td>
</tr>
</tbody>
</table>
3.2.3. Logging

Camel 4 has upgraded logging facade API `slf4j-api` from 1.7 to 2.0.

3.2.4. JUnit 4

All the `camel-test` modules that were JUnit 4.x based has been removed. All test modules now use JUnit 5.

3.2.5. API Changes

Following APIs are deprecated and removed from version 4:

- The `org.apache.camel.ExchangePattern` has removed `InOptionalOut`.
- Removed `getEndpointMap()` method from `CamelContext`.
- Removed `@FallbackConverter` as you should use `@Converter(fallback = true)` instead.
- Removed `uri` attribute on `@EndpointInject`, `@Produce`, and `@Consume` as you should use `value` (default) instead. For example, `@Produce(uri = "kafka:cheese")` should be changed to `@Produce("kafka:cheese")`
- Removed `label` on `@UriEndpoint` as you should use `category` instead.
- Removed all `asyncCallback` methods on `ProducerTemplate`. Use `asyncSend` or `asyncRequest` instead.

<table>
<thead>
<tr>
<th>Component</th>
<th>Alternative component(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>camel-swagger-java</td>
<td>camel-openapi-java</td>
</tr>
<tr>
<td>camel-websocket</td>
<td>camel-vertx-websocket</td>
</tr>
<tr>
<td>camel-websocket-jsr356</td>
<td>camel-vertx-websocket</td>
</tr>
<tr>
<td>camel-vertx-kafka</td>
<td>camel-kafka</td>
</tr>
<tr>
<td>camel-vm</td>
<td>camel-seda</td>
</tr>
<tr>
<td>camel-weka</td>
<td>none</td>
</tr>
<tr>
<td>camel-xstream</td>
<td>camel-jacksonxml</td>
</tr>
<tr>
<td>camel-zipkin</td>
<td>camel-micrometer, camel-opentelemetry</td>
</tr>
</tbody>
</table>
• Decoupled the `org.apache.camel.ExtendedCamelContext` from the `org.apache.camel.CamelContext`.

• Replaced `adapt()` from `org.apache.camel.CamelContext` with `getCamelContextExtension`.

• Decoupled the `org.apache.camel.ExtendedExchange` from the `org.apache.camel.Exchange`.

• Replaced `adapt()` from `org.apache.camel.ExtendedExchange` with `getExchangeExtension`.

• Exchange failure handling status has moved from being a property defined as `ExchangePropertyKey.FAILURE_HANDLED` to a member of the `ExtendedExchange`, accessible via `isFailureHandled()` method.

• Removed `Discard` and `DiscardOldest` from `org.apache.camel.util.concurrent.ThreadPoolRejectedPolicy`.

• Removed `org.apache.camel.builder.SimpleBuilder`. Was mostly used internally in Camel with the Java DSL in some situations.

• Moved `org.apache.camel.support.IntrospectionSupport` to `camel-core-engine` for internal use only. End users should use `org.apache.camel.spi.BeanInspection` instead.

• Removed `archetypeCatalogAsXml` method from `org.apache.camel.catalog.CamelCatalog`.

• The `org.apache.camel.health.HealthCheck` method `isLiveness` is now default `false` instead of `true`.

• Added `position` method to `org.apache.camel.StreamCache`.

• The method `configure` from the interface `org.apache.camel.main.Listener` was removed.

• The `org.apache.camel.support.EventNotifierSupport` abstract class now implements `CamelContextAware`.

• The type for `dumpRoutes` on `CamelContext` has changed from `boolean` to `String` to allow specifying either xml or yaml.

**NOTE**

The `org.apache.camel.support.PluginHelper` gives easy access to various extensions and context plugins, that was available previously in Camel v3 directly from `CamelContext`.

### 3.2.6. EIP Changes

• Removed `lang` attribute for the `<description>` on every EIPs.

• The `InOnly` and `InOut` EIPs has been removed. Instead, use `SetExchangePattern` or `To` where you can specify exchange pattern to use.

#### 3.2.6.1. Poll Enrich EIP

The polled endpoint URI is now stored as property on the `Exchange` (with key `CamelToEndpoint`) like all other EIPs. Before the URI was stored as a message header.
3.2.6.2. CircuitBreaker EIP

The following options in camel-resilience4j was mistakenly not defined as attributes:

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>bulkheadEnabled</td>
</tr>
<tr>
<td>bulkheadMaxConcurrentCalls</td>
</tr>
<tr>
<td>bulkheadMaxWaitDuration</td>
</tr>
<tr>
<td>timeoutEnabled</td>
</tr>
<tr>
<td>timeoutExecutorService</td>
</tr>
<tr>
<td>timeoutDuration</td>
</tr>
<tr>
<td>timeoutCancelRunningFuture</td>
</tr>
</tbody>
</table>

These options were not exposed in YAML DSL, and in XML DSL you need to migrate from:

```xml
< circuitBreaker >
  < resilience4jConfiguration >
    < timeoutEnabled > true </ timeoutEnabled >
    < timeoutDuration > 2000 </ timeoutDuration >
  </ resilience4jConfiguration >
  ...
</ circuitBreaker >
```

To use following attributes instead:

```xml
< circuitBreaker >
  < resilience4jConfiguration timeoutEnabled="true" timeoutDuration="2000" />
  ...
</ circuitBreaker >
```

3.2.7. XML DSL

The `<description>` to set a description on a route or node, has been changed from an element to an attribute.

Example

```xml
< route id="myRoute" description="Something that this route do" >
  < from uri="kafka:cheese" />
  ...
</ route >
```

3.2.8. Type Converter
The `String → java.io.File` converter has been removed.

### 3.2.9. Tracing

The Tracer and Backlog Tracer no longer includes internal tracing events from routes that was created by Rest DSL or route templates or Kamelets. You can turn this on, by setting `traceTemplates=true` in the tracer.

The Backlog Tracer has been enhanced and fixed to trace message headers (also streaming types). This means that previously headers of type `InputStream` was not traced before, but is now included. This could mean that the header stream is positioned at end, and logging the header afterward, may appear as the header value is empty.

### 3.2.10. UseOriginalMessage / UseOriginalBody

When `useOriginalMessage` or `useOriginalBody` is enabled in `OnException`, `OnCompletion` or error handlers, then the original message body is defensively copied and if possible converted to `StreamCache` to ensure the body can be re-read when accessed. Previously the original body was not converted to `StreamCache` which could lead to the body not able to be read or the stream has been closed.

### 3.2.11. Camel Health

Health checks are now by default only readiness checks out of the box. Camel provides the `CamelContextCheck` as both readiness and liveness checks, so there is at least one of each out of the box. Only consumer based health-checks is enabled by default.

#### 3.2.11.1. Producer Health Checks

The option `camel.health.components-enabled` has been renamed to `camel.health.producers-enabled`.

Some components (in particular AWS) provides also health checks for producers; in Camel 3.x these health checks did not work properly and has been disabled in the source. To continue this behaviour in Camel 4, the producer based health checks are disabled.

Notice that `camel-kafka` comes with producer based health-check that worked in Camel 3, and therefore this change in Camel 4, means that this health-check is disabled.

You **MUST** enable producer health-checks globally, such as in `application.properties`:

```properties
camel.health.producers-enabled = true
```

### 3.2.12. JMX

Camel now also include MBeans for `doCatch` and `doFinally` in the tree of processor MBeans.

The `ManagedChoiceMBean` have renamed `choiceStatistics` to `extendedInformation`. The `ManagedFailoverLoadBalancerMBean` have renamed `exceptionStatistics` to `extendedInformation`.

The `CamelContextMBean` and `CamelRouteMBean` has removed method `dumpRouteAsXml(boolean resolvePlaceholders, boolean resolveDelegateEndpoints)`.

### 3.2.13. YAML DSL
The backwards compatible mode Camel 3.14 or older, which allowed to have steps as child to `route` has been removed.

The new syntax is:

```
- route:
  from:
    uri: "direct:info"
  steps:
    - log: "message"
```

### 3.2.14. Backlog Tracing

The option `backlogTracing=true` is now automatically enabled to start the tracer on startup. In the previous versions the tracer was only made available, and had to be manually enabled afterwards. The old behavior can be archived by setting `backlogTracingStandby=true`.


The `org.apache.camel.impl.debugger.DefaultBacklogTracerEventMessage` has been refactored into an interface `org.apache.camel.spi.BacklogTracerEventMessage` with some additional details about traced messages. For example Camel now captures a first and last trace that contains the input and outgoing (if `InOut`) messages.

### 3.2.15. XML serialization

The default xml serialization using `ModelToXMLDumper` has been improved and now uses a generated xml serializer located in the `camel-xml-io` module instead of the JAXB based one from `camel-jaxb`.

### 3.2.16. OpenAPI Maven Plugin

The `camel-restdsl-openapi-plugin` Maven plugin now uses `platform-http` as the default rest component in the generated Rest DSL code, as it is a better default that works out of the box with Quarkus.

### 3.2.17. Component changes

#### 3.2.17.1. Category

The number of enums for `org.apache.camel.Category` has been reduced from 83 to 37, which means custom components that are using removed values need to choose one of the remainder values. We have done this to consolidate the number of categories of all components in the Camel community.

#### 3.2.17.2. camel-openapi-rest-dsl-generator

This dsl-generator has updated the underlying model classes (`apicurio-data-models`) from 1.1.27 to 2.0.3.

#### 3.2.17.3. camel-atom

The `camel-atom` component has changed the 3rd party atom client from Apache Abdera to RSSReader.
The camel-atom component has changed the 3rd party atom client from Apache Abdera to RSSReader. This means the feed object is changed from org.apache.abdera.model.Feed to com.apptasticsoftware.rssreader.Item.

3.2.17.4. camel-azure-cosmosdb

The itemPartitionKey has been updated. It’s now a String a not a PartitionKey anymore. More details in CAMEL-19222.

3.2.17.5. camel-bean

When using the method option to refer to a specific method, and using parameter types and values, such as: "bean:myBean?method=foo(com.foo.MyOrder, true)" then any class types must now be using .class syntax, i.e. com.foo.MyOrder should now be com.foo.MyOrder.class.

Example

"bean:myBean?method=foo(com.foo.MyOrder.class, true)"

This also applies to Java types such as String, int.

"bean:myBean?method=bar(String.class, int.class)"

3.2.17.6. camel-box

Upgraded from Box Java SDK v2 to v4, which have some method signature changes. The method to get a file thumbnail is no longer available.

3.2.17.7. camel-caffeine

The keyType parameter has been removed. The Key for the cache will now be only String type. More information in CAMEL-18877.

3.2.17.8. camel-fhir

The underlying hapi-fhir library has been upgraded from 4.2.0 to 6.2.4. Only the Delete API method has changed and now returns ca.uhn.fhir.rest.api.MethodOutcome instead of org.hl7.fhir.instance.model.api.IBaseOperationOutcome. See hapi-fhir for a more detailed list of underlying changes (only the hapi-fhir client is used in Camel).

3.2.17.9. camel-google

The API based components camel-google-drive, camel-google-calendar, camel-google-sheets and camel-google-mail has been upgraded from Google Java SDK v1 to v2 and to latest API revisions. The camel-google-drive and camel-google-sheets have some API methods changes, but the others are identical as before.

3.2.17.10. camel-http

The component has been upgraded to use Apache HttpComponents v5 which has an impact on how the underlying client is configured. There are 4 different timeouts (connectionRequestTimeout, connectTimeout, soTimeout and responseTimeout) instead of initially 3.
(connectionRequestTimeout, connectTimeout and socketTimeout) and the default value of some of them has changed so please refer to the documentation for more details.

Please note that the socketTimeout has been removed from the possible configuration parameters of HttpClient, use responseTimeout instead.

Finally, the option soTimeout along with any parameters included into SocketConfig, need to be prefixed by httpConnection, the rest of the parameters including those defined into HttpClientBuilder and RequestConfig still need to be prefixed by httpClient, like before.

3.2.17.11. camel-http-common

The API in org.apache.camel.http.common.HttpBinding has changed slightly to be more reusable. The parseBody method now takes in HttpServletRequest as input parameter. And all HttpMessage has been changed to generic Message types.

3.2.17.12. camel-kubernetes

The io.fabric8:kubernetes-client library has been upgraded and some deprecated API usage has been removed. Operations previously prefixed with replace are now prefixed with update.

For example replaceConfigMap is now updateConfigMap, replacePod is now updatePod etc. The corresponding constants in class KubernetesOperations are also renamed. REPLACE_CONFIGMAP_OPERATION is now UPDATE_CONFIGMAP_OPERATION, REPLACE_POD_OPERATION is now UPDATE_POD_OPERATION etc.

3.2.17.13. camel-web3j

The camel-web3j has upgrade web3j JAR from 3.x to 5.0 which has many API changes, and so some previous API calls are no long provided.

3.2.17.14. camel-main

The following constants have been moved from BaseMainSupport / Main to MainConstants:

<table>
<thead>
<tr>
<th>Old Name</th>
<th>New Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main.DEFAULTPROPERTY_PLACEHOLDER_LOCATION</td>
<td>MainConstants.DEFAULTPROPERTY_PLACEHOLDER_LOCATION</td>
</tr>
<tr>
<td>Main.INITIAL_PROPERTIES_LOCATION</td>
<td>MainConstants.INITIAL_PROPERTIES_LOCATION</td>
</tr>
<tr>
<td>Main.OVERRIDE_PROPERTIES_LOCATION</td>
<td>MainConstants.OVERRIDE_PROPERTIES_LOCATION</td>
</tr>
<tr>
<td>Main.PROPERTY_PLACEHOLDER_LOCATION</td>
<td>MainConstants.PROPERTY_PLACEHOLDER_LOCATION</td>
</tr>
</tbody>
</table>

3.2.17.15. camel-micrometer

The metrics has been renamed to follow Micrometer naming convention.
### 3.2.17.16. camel-jbang

The command `camel dependencies` has been renamed to `camel dependency`.

In Camel JBang the `-dir` parameter for `init` and `run` goal has been renamed to require 2 dashes `--dir` like all the other options.

The `camel stop` command will now by default stop all running integrations (the option `--all` has been removed).

The `Placeholders substitutes` is changed to use `#name` instead of `$name` syntax.

### 3.2.17.17. camel-openapi-java

The `camel-openapi-java` component has been changed to use `io.swagger.v3` libraries instead of `io.apicurio.datamodels`. As a result, the return type of the public method `org.apache.camel.openapi.RestOpenApiReader.read()` is now `io.swagger.v3.oas.models.OpenAPI` instead of `io.apicurio.datamodels.openapi.models.OasDocument`. When an OpenAPI 2.0 (swagger) specification is parsed, it is automatically upgraded to OpenAPI 3.0.x by the swagger parser. This version also supports OpenAPI 3.1.x specifications.

### 3.2.17.18. camel-optaplanner

The `camel-optaplanner` component has been change to use `SolverManager`. If you were using `SoverManager` in Camel 3, you don’t need anymore the boolean `useSolverManager` in the Route. Deprecated `ProblemFactChange` has been replaced by `ProblemChange`.

---

<table>
<thead>
<tr>
<th>Old Name</th>
<th>New Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CamelExchangeEventNotifier</td>
<td>camel.exchange.event.notifier</td>
</tr>
<tr>
<td>CamelExchangesFailed</td>
<td>camel.exchanges.failed</td>
</tr>
<tr>
<td>CamelExchangesFailuresHandled</td>
<td>camel.exchanges.failures.handled</td>
</tr>
<tr>
<td>CamelExchangesInflight</td>
<td>camel.exchanges.external.redeliveries</td>
</tr>
<tr>
<td>CamelExchangesSucceeded</td>
<td>camel.exchanges.succeeded</td>
</tr>
<tr>
<td>CamelExchangesTotal</td>
<td>camel.exchanges.total</td>
</tr>
<tr>
<td>CamelMessageHistory</td>
<td>camel.message.history</td>
</tr>
<tr>
<td>CamelRoutePolicy</td>
<td>camel.route.policy</td>
</tr>
<tr>
<td>CamelRoutePolicyLongTask</td>
<td>camel.route.policy.long.task</td>
</tr>
<tr>
<td>CamelRoutesAdded</td>
<td>camel.routes.added</td>
</tr>
<tr>
<td>CamelRoutesRunning</td>
<td>camel.routes.running</td>
</tr>
</tbody>
</table>

Red Hat build of Apache Camel 4.0 Migrating Fuse 7 Applications to Red Hat build of Apache Camel for Quarkus
The new URI path is:

```java
from("optaplanner:myProblemName")
.to("...")
```

You can pass the Optaplanner SolverManager in 2 ways:

- as #parameter
- as header

When running `camel-optaplanner` on Quarkus, use the Quarkus way of creating the SolverManager.

You can migrate legacy Camel Optaplanner Routes, which will allow Camel Optaplanner to handle creating the SolverManager for those legacy Routes, by providing the XML config file, as show in the code below:

**Providing Optaplanner Routes XML config file**

```java
from("optaplanner:myProblemName?configFile=PATH/TO/CONFIG.FILE.xml")
.to("...")
```

**NOTE**

Solver Daemon solutions should be migrated to use SolverManager.

### 3.2.17.19. camel-platform-http-vertx

If the route or consumer is suspended then http status 503 is now returned instead of 404.

### 3.2.17.20. camel-salesforce

Property names of blob fields on generated DTOs no longer have 'Url' affixed. For example, the `ContentVersionUrl` property is now `ContentVersion`.

### 3.2.17.21. camel-slack

The default delay (on slack consumer) is changed from 0.5s to 10s to avoid being rate limited to often by Slack.

### 3.2.17.22. camel-micrometer-starter

The `uri` tags are now static instead of dynamic (by default), as potential too many tags generated due to URI with dynamic values. This can be enabled again by setting `camel.metrics.uriTagDynamic=true`.

### 3.2.17.23. camel-platform-http-starter

The `platform-http-starter` has been changed from using `camel-servlet` to use the HTTP server directly. Therefore, all the HTTP endpoints are no longer prefixed with the servlet context-path (default is `camel`).

For example:

**HTTP endpoint**

```java
```
The endpoint can be called with http://localhost:8080/myservice, as the context-path is not in use.

NOTE

The platform-http-starter can also be used with Rest DSL.

If the route or consumer is suspended then http status 503 is now returned instead of 404.

3.2.17.24. camel-twitter

The camel-twitter component was updated to use Twitter4j version 4.1.2, which has moved the packages used by a few of its classes. If accessing certain twitter-related data, such as the Twit status, you need to update the packages used from twitter4j.Status to twitter4j.v1.Status.

3.3. MIGRATING TO APACHE CAMEL 3

This guide provides information on migrating from Red Hat Fuse 7 to Camel 3

NOTE

There are important differences between Fuse 7 and Camel 3 in the components, such as modularization and XML Schema changes. See each component section for details.

Red Hat build of Apache Camel for Quarkus supports Camel version 4. This section provides information relating to upgrading Camel when you migrate your Red Hat Fuse 7 application to Red Hat build of Apache Camel for Quarkus with Camel version 3.

3.3.1. Java versions

Camel 3 supports Java 17 and Java 11 but not Java 8.

3.3.1.1. JAXB removed in JDK 11

In Java 11 the JAXB modules have been removed from the JDK, therefore you will need to add them as Maven dependencies (if you use JAXB such as when using XML DSL or the camel-jaxb component):

```xml
<dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.1</version>
</dependency>

<dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-core</artifactId>
    <version>2.3.0.1</version>
</dependency>
```
NOTE

The Java Platform Standard Edition 11 Development Kit (JDK 11) is deprecated in release version Camel 3.x and is not supported in release versions 4.x.

3.3.2. Modularization of camel-core

In Camel 3.x, camel-core has been split into many JARs as follows:

- camel-api
- camel-base
- camel-caffeine-lrucache
- camel-cloud
- camel-core
- camel-jaxp
- camel-main
- camel-management-api
- camel-management
- camel-support
- camel-util
- camel-util-json

Maven users of Apache Camel can keep using the dependency camel-core which has transitive dependencies on all of its modules, except for camel-main, and therefore no migration is needed.

3.3.3. Modularization of Components

In Camel 3.x, some of the camel-core components are moved into individual components.

- camel-attachments
- camel-bean
- camel-browse
- camel-controlbus
- camel-dataformat
- camel-dataset
3.3.4. Multiple CamelContexts per application not supported

Support for multiple CamelContexts has been removed and only one CamelContext per deployment is recommended and supported. The context attribute on the various Camel annotations such as @EndpointInject, @Produce, @Consume etc. has therefore been removed.

3.3.5. Deprecated APIs and Components

All deprecated APIs and components from Camel 2.x have been removed in Camel 3.

3.3.5.1. Removed components

All deprecated components from Camel 2.x are removed in Camel 3.x:

- camel-http
- camel-hdfs
- camel-mina
- camel-mongodb
- camel-netty
- camel-netty-http
- camel-quartz
- camel-restlet
- camel-rx
- camel-jibx
- camel-boon dataformat
- camel-linkedln

The Linkedin API is no longer supported.

- camel-zookeeper
The component route policy functionality is removed. Use \texttt{ZooKeeperClusterService} or the \texttt{camel-zookeeper-master} instead.

camel-jetty

No longer supports \texttt{producer} (which has been removed). Use \texttt{camel-http} component instead.

twitter-streaming

Removed as it relied on the deprecated Twitter Streaming API and is no longer functional.

3.3.5.2. Renamed components

The following components are renamed in Camel 3.x.

camel-microprofile-metrics

Renamed to \texttt{camel-micrometer}

test

Renamed to \texttt{dataset-test} and moved out of \texttt{camel-core} into \texttt{camel-dataset} JAR.

http4

Renamed to \texttt{http}, and it’s corresponding component package from
\texttt{org.apache.camel.component.http4} to \texttt{org.apache.camel.component.http}. The supported schemes are now only \texttt{http} and \texttt{https}.

hdfs2

Renamed to \texttt{hdfs}, and it’s corresponding component package from
\texttt{org.apache.camel.component.hdfs2} to \texttt{org.apache.camel.component.hdfs}. The supported scheme is now \texttt{hdfs}.

mina2

Renamed to \texttt{mina}, and it’s corresponding component package from
\texttt{org.apache.camel.component.mina2} to \texttt{org.apache.camel.component.mina}. The supported scheme is now \texttt{mina}.

mongodb3

Renamed to \texttt{mongodb}, and it’s corresponding component package from
\texttt{org.apache.camel.component.mongodb3} to \texttt{org.apache.camel.component.mongodb}. The supported scheme is now \texttt{mongodb}.

netty4-http

been renamed to \texttt{netty-http}, and it’s corresponding component package from
\texttt{org.apache.camel.component.netty4.http} to \texttt{org.apache.camel.component.netty.http}. The supported scheme is now \texttt{netty-http}.

netty4

Renamed to \texttt{netty}, and it’s corresponding component package from
\texttt{org.apache.camel.component.netty4} to \texttt{org.apache.camel.component.netty}. The supported scheme is now \texttt{netty}.

quartz2

Renamed to \texttt{quartz}, and it’s corresponding component package from
\texttt{org.apache.camel.component.quartz2} to \texttt{org.apache.camel.component.quartz}. The supported scheme is now \texttt{quartz}.

rxjava2

Renamed to \texttt{rxjava}, and it’s corresponding component package from
\texttt{org.apache.camel.component.rxjava2} to \texttt{org.apache.camel.component.rxjava}.

camel-jetty9
Renamed to camel-jetty. The supported scheme is now jetty.

3.3.6. Changes to Camel components

3.3.6.1. Mock component

The mock component has been moved out of camel-core. Because of this a number of methods on its assertion clause builder are removed.

3.3.6.2. ActiveMQ

If you are using the activemq-camel component, then you should migrate to use camel-activemq component, where the component name has changed from org.apache.activemq.camel.component.ActiveMQComponent to org.apache.camel.component.activemq.ActiveMQComponent.

3.3.6.3. AWS

The component camel-aws has been split into multiple components:

- camel-aws-cw
- camel-aws-ddb (which contains both ddb and ddbstreams components)
- camel-aws-ec2
- camel-aws-iam
- camel-aws-kinesis (which contains both kinesis and kinesis-firehose components)
- camel-aws-kms
- camel-aws-lambda
- camel-aws-mq
- camel-aws-s3
- camel-aws-sdb
- camel-aws-ses
- camel-aws-sns
- camel-aws-sqs
- camel-aws-swf

**NOTE**

It is recommended to add specific dependencies for these components.

3.3.6.4. Camel CXF
The camel-cxf JAR has been divided into SOAP vs REST. We recommended you choose the specific JAR from the following list when migrating from camel-cxf.

- camel-cxf-soap
- camel-cxf-rest
- camel-cxf-transport

For example, if you were using CXF for SOAP, then select camel-cxf-soap and camel-cxf-transport when migrating from camel-cxf.

### 3.3.6.4.1. Camel CXF changed namespaces

The camel-cxf XML XSD schemas has also changed namespaces.

#### Table 3.1. Changes to namespaces

<table>
<thead>
<tr>
<th>Old Namespace</th>
<th>New Namespace</th>
</tr>
</thead>
</table>

The camel-cxf SOAP component is moved to a new jaxws sub-package, that is, org.apache.camel.component.cxf is now org.apache.camel.component.cxf.jaxws. For example, the CxfComponent class is now located in org.apache.camel.component.cxf.jaxws.

### 3.3.6.5. FHIR

The camel-fhir component has upgraded it’s hapi-fhir dependency to 4.1.0. The default FHIR version has been changed to R4. Therefore, if DSTU3 is desired it has to be explicitly set.

### 3.3.6.6. Kafka

The camel-kafka component has removed the options bridgeEndpoint and circularTopicDetection as this is no longer needed as the component is acting as bridging would work on Camel 2.x. In other words camel-kafka will send messages to the topic from the endpoint uri. To override this use the KafkaConstants.OVERRIDE_TOPIC header with the new topic. See more details in the camel-kafka component documentation.

### 3.3.6.7. Telegram

The camel-telegram component has moved the authorization token from uri-path to a query parameter instead, e.g. migrate

```
telegram:bots/myTokenHere
```
3.3.6.8. JMX

If you run Camel standalone with just camel-core as a dependency, and you want JMX enabled out of the box, then you need to add camel-management as a dependency.

For using ManagedCamelContext you now need to get this extension from CamelContext as follows:

```java
ManagedCamelContext managed = camelContext.getExtension(ManagedCamelContext.class);
```

3.3.6.9. XSLT

The XSLT component has moved out of camel-core into camel-xslt and camel-xslt-saxon. The component is separated so camel-xslt is for using the JDK XSTL engine (Xalan), and camel-xslt-saxon is when you use Saxon. This means that you should use xslt and xslt-saxon as component name in your Camel endpoint URIs. If you are using XSTL aggregation strategy, then use org.apache.camel.component.xslt.saxon.XsltSaxonAggregationStrategy for Saxon support. And use org.apache.camel.component.xslt.saxon.XsltSaxonBuilder for Saxon support if using xslt builder. Also notice that allowStax is also only supported in camel-xslt-saxon as this is not supported by the JDK XSLT.

3.3.6.10. XML DSL Migration

The XML DSL has been changed slightly.

The custom load balancer EIP has changed from `<custom>` to `<customLoadBalancer>`

The XMLSecurity data format has renamed the attribute keyOrTrustStoreParametersId to keyOrTrustStoreParametersRef in the `<secureXML>` tag.

The `<zipFile>` data format has been renamed to `< zipfile>.

3.3.7. Migrating Camel Maven Plugins

The camel-maven-plugin has been split up into two maven plugins:

**camel-maven-plugin**

camel-maven-plugin has the **run** goal, which is intended for quickly running Camel applications standalone. See https://camel.apache.org/manual/camel-maven-plugin.html for more information.

**camel-report-maven-plugin**

The camel-report-maven-plugin has the **validate** and **route-coverage** goals which is used for generating reports of your Camel projects such as validating Camel endpoint URIs and route coverage reports, etc. See https://camel.apache.org/manual/camel-report-maven-plugin.html for more information.
CHAPTER 4. MIGRATING CAMEL QUARKUS PROJECTS

4.1. UPDATING PROJECTS TO THE LATEST QUARKUS VERSION

We recommend that you use Maven to update and upgrade your projects to the latest Quarkus version.

- Section 4.1.2, “Updating with Maven”

IMPORTANT

For projects that use Hibernate ORM or Hibernate Reactive, review the Hibernate ORM 5 to 6 migration quick reference. The following update command covers only a subset of this guide.

4.1.1. Prerequisites

- Roughly 30 minutes
- JDK installed with JAVA_HOME configured appropriately
- Apache Maven 3.8.6
- Optionally, the Quarkus CLI if you want to use it
- A project based on Camel Quarkus version 2.13 or later.

4.1.2. Updating with Maven

1. Configure your extension registry client as described in the Configuring Quarkus extension registry client section of the Quarkus Getting Started guide.

2. Update with Maven:
   Go to the project directory and update the project to the latest stream:
   a. Ensure that the Quarkus Maven plugin version aligns with the latest supported Red Hat build of Quarkus version.
   b. Run the update with the following command:

```
mvn io.quarkus.platform:quarkus-maven-plugin:3.5.2:update -N
```

For multi-module projects use following command

```
find . -type f -name "pom.xml" -execdir sh -c "mvn io.quarkus.platform:quarkus-maven-plugin:3.5.2:update -N" \
```

Optional

By default, this command updates to the latest current version. To update to a specific stream instead of latest current version, add the stream option to this command followed by the version; for example: `--Dstream=3.2`
NOTE

Updates of multi-module project may show a lot of errors, because the update tool fails to update modules with

```
<packaging>pom</packaging>
```

If such modules are present (typically containing versions), update them manually.

3. Analyze the update command output for potential instructions and perform the suggested tasks if needed.

4. Use a diff tool to inspect all changes.

5. Review the migration guide for items that were not updated by the update command. If your project has such items, implement the additional steps advised in these topics.

6. Ensure the project builds without errors, all tests pass, and the application functions as required before deploying to production.

7. Before deploying your updated Quarkus application to production, ensure the following:
   - The project builds without errors.
   - All tests pass.
   - The application functions as required.
CHAPTER 5. ADDITIONAL RESOURCES

For more information about Red Hat build of Apache Camel for Quarkus, see the following documentation:

- Red Hat build of Apache Camel for Quarkus Reference
- Getting Started with Red Hat build of Apache Camel for Quarkus
- Developing Applications with Red Hat build of Apache Camel for Quarkus
- Migrating applications to Red Hat build of Quarkus