Red Hat build of Apache Camel Extensions for Quarkus 2.13

Migrating Fuse 7 Applications to Camel Extensions for Quarkus

Migrating Fuse 7 Applications to Camel Extensions for Quarkus provided by Red Hat

Last Updated: 2023-06-22

Red Hat build of Apache Camel Extensions for Quarkus 2.13 Migrating Fuse 7 Applications to Camel Extensions for Quarkus

Migrating Fuse 7 Applications to Camel Extensions for Quarkus provided by Red Hat

Abstract

Migrating Fuse 7 Applications to Camel Extensions for Quarkus provides information on migrating from Red Hat Fuse 7 to Red Hat build of Camel Extensions for Quarkus.

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. OVERVIEW OF MIGRATING FUSE 7 APPLICATIONS TO CAMEL EXTENSIONS 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)

Camel Extensions for Quarkus

Camel Extensions for Quarkus (CEQ) 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 CEQ 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 Extensions 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.

CHAPTER 2. MIGRATING CAMEL ROUTES FROM FUSE 7 TO CAMEL EXTENSIONS FOR QUARKUS (CEQ)

NOTE

You can define Camel routes in CEQ 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 CEQ application and add the necessary dependencies to your CEQ 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:

```
$ cd <directory_name>
```

3. Create a file named **Routes.java** in the **src/main/java/org/acme/** subfolder.
4. Add the route definition from your Fuse application to the **Routes.java**, similar to the following example:

```
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 CEQ application. You will then need to add the necessary dependencies to the CEQ **pom.xml** file and update your CEQ configuration in the **application.properties** file.

NOTE

CEQ supports Camel 3, whereas Fuse 7 supports Camel 2. For more information relating to upgrading Camel when you migrate your Red Hat Fuse 7 application to CEQ, see Migrating from Camel 2 to Camel 3 .

For more information about using beans in Camel Quarkus, see the CDI and the Camel Bean Component section in the *Developing Applications with Camel Extensions 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 **<camelContext>** 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:

```
<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 uri="/">
                <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

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

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

```
import org.apache.camel.builder.RouteBuilder;
public class Routes extends RouteBuilder {
    public void configure() {
      restConfiguration()
           .contextPath("/camel");
    }
}
```

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 Camel Extensions 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>
```

6. Create a file named **camel-routes.xml** in the **src/main/resources/routes/** directory.
7. Copy the **<route>** element and sub-elements from the following **blueprint-example.xml** example to the **camel-routes.xml** file:

blueprint-example.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" uri="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"/>
               <to id="_to1" uri="file:work/cbr/output/uk"/>
            </when>
            <when id="_when2">
               <xpath id="_xpath2">/order/customer/country = 'US'</xpath>
               <log id="_log3" message="Sending order ${file:name} to the US"/>
               <to id="_to2" uri="file:work/cbr/output/us"/>
            </when>
            <otherwise id="_otherwise1">
```

```
                <log id="_log4" message="Sending order ${file:name} to another country"/>
                <to id="_to3" uri="file:work/cbr/output/others"/>
            </otherwise>
        </choice>
        <log id="_log5" message="Done processing ${file:name}"/>
    </route>
  </camelContext>
</blueprint>
```

camel-routes.xml

```
<route id="cbr-route">
  <from id="_from1" uri="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"/>
      <to id="_to1" uri="file:work/cbr/output/uk"/>
    </when>
    <when id="_when2">
      <xpath id="_xpath2">/order/customer/country = 'US'</xpath>
      <log id="_log3" message="Sending order ${file:name} to the US"/>
      <to id="_to2" uri="file:work/cbr/output/us"/>
    </when>
    <otherwise id="_otherwise1">
      <log id="_log4" message="Sending order ${file:name} to another country"/>
      <to id="_to3" uri="file:work/cbr/output/others"/>
    </otherwise>
  </choice>
  <log id="_log5" message="Done processing ${file:name}"/>
</route>
```

8. Modify **application.properties**

```
# Camel
#
camel.context.name = camel-quarkus-xml-io-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.

2.3. ADDITIONAL RESOURCES

For more information about Camel Extensions for Quarkus (CEQ), see the following documentation:

- Camel Extensions for Quarkus Reference
- Getting Started with Camel Extensions for Quarkus
- Developing Applications with Camel Extensions for Quarkus

CHAPTER 3. MIGRATING FROM CAMEL 2 TO CAMEL 3

Camel Extensions for Quarkus supports Camel version 3 whereas Fuse 7 supported Camel version 2. This section provides information relating to upgrading Camel when you migrate your Red Hat Fuse 7 application to Camel Extensions for Quarkus.

3.1. JAVA VERSIONS

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

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. 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
- camel-direct
- camel-directvm
- camel-file
- camel-language
- camel-log
- camel-mock
- camel-ref
- camel-rest
- camel-saga
- camel-scheduler
- camel-seda
- camel-stub
- camel-timer
- camel-validator
- camel-vm
- camel-xpath
- camel-xslt
- camel-xslt-saxon
- camel-zip-deflater

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.

3.5. DEPRECATED APIS AND COMPONENTS

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

3.5.1. Removed components

All deprecated components from Camel 2.x are removed in Camel 3.x, including the old **camel-http**, **camel-hdfs**, **camel-mina**, **camel-mongodb**, **camel-netty**, **camel-netty-http**, **camel-quartz**, **camel-restlet** and **camel-rx** components.

- Removed **camel-jibx** component.
- Removed **camel-boon** dataformat.
- Removed the **camel-linkedin** component as the Linkedin API 1.0 is no longer supported. Support for the new 2.0 API is tracked by CAMEL-13813.
- The **camel-zookeeper** has its route policy functionality removed, instead use **ZooKeeperClusterService** or the **camel-zookeeper-master** component.
- The **camel-jetty** component no longer supports producer (which has been removed), use **camel-http** component instead.
- The **twitter-streaming** component has been removed as it relied on the deprecated Twitter Streaming API and is no longer functional.

3.5.2. Renamed components

Following components are renamed in Camel 3.x.

- The **Camel-microprofile-metrics** has been renamed to **camel-micrometer**
- The **test** component has been renamed to **dataset-test** and moved out of **camel-core** into **camel-dataset** JAR.
- The **http4** component has been renamed to **http**, and it's corresponding component package from **org.apache.camel.component.http4** to **org.apache.camel.component.http**. The supported schemes are now only **http** and **https**.
- The **hdfs2** component has been renamed to **hdfs**, and it's corresponding component package from **org.apache.camel.component.hdfs2** to **org.apache.camel.component.hdfs**. The supported scheme is now **hdfs**.
- The **mina2** component has been renamed to **mina**, and it's corresponding component package from **org.apache.camel.component.mina2** to **org.apache.camel.component.mina**. The supported scheme is now **mina**.
- The **mongodb3** component has been renamed to **mongodb**, and it's corresponding component package from **org.apache.camel.component.mongodb3** to **org.apache.camel.component.mongodb**. The supported scheme is now **mongodb**.
- The **netty4-http** component has been renamed to **netty-http**, and it's corresponding component package from **org.apache.camel.component.netty4.http** to **org.apache.camel.component.netty.http**. The supported scheme is now **netty-http**.
- The **netty4** component has been renamed to **netty**, and it's corresponding component package from **org.apache.camel.component.netty4** to **org.apache.camel.component.netty**. The supported scheme is now **netty**.
- The **quartz2** component has been renamed to **quartz**, and it's corresponding component package from **org.apache.camel.component.quartz2** to **org.apache.camel.component.quartz**. The supported scheme is now **quartz**.
- The **rxjava2** component has been renamed to **rxjava**, and it's corresponding component package from **org.apache.camel.component.rxjava2** to **org.apache.camel.component.rxjava**.

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

3.6. CHANGES TO CAMEL COMPONENTS

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.

3.6.2. 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**.

3.6.3. 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 specifc dependencies for these components.

3.6.4. Camel CXF

The **camel-cxf** JAR has been divided into SOAP vs REST and Spring and non Spring JARs. It is recommended to choose the specific JAR from the following list when migrating from **came-cxf**.

- **camel-cxf-soap**
- **camel-cxf-spring-soap**
- **camel-cxf-rest**
- **camel-cxf-spring-rest**
- **camel-cxf-transport**
- **camel-cxf-spring-transport**

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

When using Spring Boot, choose from the following starter when you migrate from **camel-cxf-starter** to SOAP or REST:

- **camel-cxf-soap-starter**
- **camel-cxf-rest-starter**

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

Table 3.1. Changes to namespaces

Old Namespace	New Namespace
http://camel.apache.org/schema/cxf	**http://camel.apache.org/schema/cxf/jaxws**
http://camel.apache.org/schema/cxf/camel-cxf.xsd	**http://camel.apache.org/schema/cxf/jaxws/camel-cxf.xsd**
http://camel.apache.org/schema/cxf	**http://camel.apache.org/schema/cxf/jaxrs**
http://camel.apache.org/schema/cxf/camel-cxf.xsd	**http://camel.apache.org/schema/cxf/jaxrs/camel-cxf.xsd**

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.jaws**. For example, the **CxfComponent** class is now located in **org.apache.camel.component.cxf.jaxws**.

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.

3.6.6. 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.

3.6.7. Telegram

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

```
telegram:bots/myTokenHere
```

to

```
telegram:bots?authorizationToken=myTokenHere
```

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:

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

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 XSLT 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.

3.6.10. 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>**.

3.7. 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.