Red Hat Fuse 7.8

Migration Guide

Migrating to Red Hat Fuse 7.8

Last Updated: 2020-12-18

Red Hat Fuse 7.8 Migration Guide

Migrating to Red Hat Fuse 7.8

Legal Notice

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Use this guide to help you when upgrading your Fuse installation to the latest version of Red Hat Fuse.

Table of Contents

PREFACE

This guide provides information on updating Red Hat Fuse and Fuse applications:

NOTE

If you want to migrate from Fuse 6 to the latest Fuse 7 release, before you follow the instructions in this guide, you should follow the instructions in the Red Hat Fuse 7.0 Migration Guide.

CHAPTER 1. UPGRADING FUSE ONLINE

The Fuse Online upgrade process depends on whether Fuse Online is installed on Red Hat OpenShift Online or on OpenShift Container Platform (OCP).

- **OpenShift Online** - When Fuse 7.8 is released, the Fuse Online infrastructure on OpenShift Online is automatically upgraded. You must republish any running integrations as described in Upgrading Fuse Online integrations that are running on OpenShift Online .
- **OCP** - To upgrade a Fuse Online environment that is running on OpenShift Container Platform on-site, you should first set up access to Docker images for the Fuse backup and restore database, as described in Access Docker images before an upgrade .
  The procedure for upgrading Fuse depends on how you installed Fuse:
  - If you used the OperatorHub to install Fuse, see Upgrading Fuse by using the OperatorHub .
  - If you used the install script to install Fuse, see Upgrading Fuse by using the install script .

  To upgrade your integrations, you should also consider Apache Camel updates as described in Camel Migration Considerations.

1.1. ACCESS DOCKER IMAGES BEFORE AN UPGRADE

By default, the Fuse Online upgrade process pulls Docker images for the Fuse Online backup and restore database from the **docker.io** registry, instead of pulling them from **registry.redhat.io** (as described in the known issue https://issues.redhat.com/browse/ENTESB-15364). Because **docker.io** imposes a service limitation on free downloads, the Fuse Online upgrade process could encounter the following error when it attempts to download the backup and restore database images:

```
error: code = Unknown desc = toomanyrequests: You have reached your pull rate limit
```

For more information about the Docker Hub pull rate limit, see the Docker documentation (https://www.docker.com/increase-rate-limits).

To avoid encountering the Docker limit error, before you start the Fuse Online upgrade process, make sure that you have access to Docker images. How you access the Docker images depends on your OpenShift Container Platform (OCP) version:

- For OCP 4.6, add Docker Hub credentials into your existing syndesis pull secret as described in Accessing Docker images on OCP 4.6. When you add your Docker credentials to the syndesis pull secret, the Docker Hub limit on API is raised, depending on your Docker Hub account type.
- For OCP 3.11, pull the Fuse Online backup and restore database images from **docker.io** to a local cache as described in Accessing Docker images on OCP 3.11.

1.1.1. Accessing Docker images on OCP4

To avoid encountering a possible Docker limit error, before you start the Fuse Online upgrade process, add your docker credentials to the syndesis pull secret.

Prerequisite

You have OpenShift cluster administrator access.

Procedure

To add Docker Hub credentials into your existing syndesis pull secret:

1. Log into OpenShift with your administrator credentials:

```
oc login -u system:admin
```

2. Retrieve your syndesis pull secret by running the following command:

```
 oc get secrets syndesis-pull-secret -o=custom-columns=SECRET:.data.* --no-headers |
base64 -d | jq
```

   This command returns output similar to the following, where <AUTH> is your encoded username and password in base64:

```
{
  "auths":{
    "registry.redhat.io":{
      "username":"<AUTH>",
      "password":"<AUTH>",
      "auth":"<AUTH>"
    },
    <You can have more auths elements here>
  }
 }
```

3. To add your Docker Hub registry credentials, edit your pull secret and save it to the PULL_SECRET variable:

```
 PULL_SECRET=$(base64 -w 0 <<EOF
{
  "auths":{
    "registry.redhat.io":{
      "username":"<AUTH>",
      "password":"<AUTH>",
      "auth":"<AUTH>"
    },
    <You can have more auths elements here>
    },
    "https://index.docker.io/v1/": {
      "auth": "<AUTH>"
    }
  }
}
EOF
)
```

4. To patch your syndesis pull secret, run the following command:

```
oc apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: syndesis-pull-secret
data:
```

```
  .dockerconfigjson: $PULL_SECRET
type: kubernetes.io/dockerconfigjson
EOF
```

Next step

- If you used the OperatorHub to install Fuse Online, see Upgrading Fuse Online by using the OperatorHub.
- If you used the install script to install Fuse Online, see Upgrading Fuse Online by using the install script.

1.1.2. Accessing Docker images on OCP 3.11

To avoid encountering a possible Docker limit error, before you start the Fuse Online upgrade process on OCP 3.11, you should pull the required images from **docker.io** to a local cache.

Prerequisite

You have OpenShift cluster administrator access.

Procedure

1. Login to the OpenShift node with your OpenShift username and the OpenShift cluster URL:

```
ssh login <user>@node1.<clusterUrl>
```

2. Login to docker with your username and password:

```
sudo docker login -u <username> -p <password> docker.io
```

3. Run the following commands that pull the PostgreSQL images:

```
sudo docker pull crunchydata/crunchy-pgdump:centos7-10.11-4.2.1
sudo docker pull crunchydata/crunchy-pgrestore:centos7-10.11-4.2.1
sudo docker pull centos:7
```

   The PostgresQL images are pulled to the OpenShift node's internal docker registry.

4. Repeat the previous three steps for each OpenShift node.

Next step

Upgrade Fuse Online by following the steps in Upgrading Fuse Online by using the install script .

1.2. UPGRADING FUSE ONLINE BY USING THE OPERATORHUB

Fuse Online 7.8 is available in the OperatorHub starting with OCP 4.6. If you are using OCP 4.5, you must upgrade to OCP 4.6 if you want to install Fuse Online 7.8.

Note: If you used the install script to install Fuse Online 7.7, you should use the install script to upgrade to Fuse Online 7.8 as described in Upgrading Fuse Online by using the install script .

When you install Fuse Online, you specify a channel with the **fuse-online-v7.n** format, where **n** is the current release number. For example, for Fuse Online 7.8, the channel is **fuse-online-v7.8**.

Upgrading from a Fuse Online 7.8 version to a newer Fuse Online 7.8 version depends on the **Approval Strategy** that you selected when you installed Fuse Online:

- For **Automatic** updates, when a new version of the Fuse Online operator is available, the OpenShift Operator Lifecycle Manager (OLM) automatically upgrades the running instance of the Fuse Online without human intervention.
- For **Manual** updates, when a newer version of an Operator is available, the OLM creates an update request. As a cluster administrator, you must then manually approve that update request to have the Fuse Online operator updated to the new version as described in the Manually approving a pending Operator upgrade section of the OpenShift documentation.

During and after an infrastructure upgrade, existing integrations continue to run with the *older* versions of Fuse Online libraries and dependencies. To have them run with the updated versions, you must republish them.

For OCP 4.6 and later, to upgrade from Fuse Online 7.7 to 7.8, use the steps in the following procedure.

Procedure

1. To avoid encountering Docker limit errors, before you start the Fuse Online upgrade process, add your docker credentials to the syndesis pull secret as described in as described in Access Docker images before an upgrade.
2. To upgrade the Fuse Online operator:
   a. In the OpenShift web console, click **Operators** > **Installed Operators**.
   b. Click the **Fuse Online** operator and then **Subscription**.
   c. Next to the **Channel**, click the **Edit** icon.
   d. Select the **fuse-online-v7.8** channel and then click **Save**.

If you specified **Manual** updates when you installed Fuse Online, approve the Operator update request by following the instructions in the Manually approving a pending Operator upgrade section of the OpenShift documentation.

1.3. UPGRADING FUSE ONLINE BY USING THE INSTALL SCRIPT

If you installed Fuse Online by using the install script (rather than the OperatorHub), here are the general steps to upgrade Fuse Online:

- A cluster administrator sets up access to Docker images for the the Fuse Online backup and restore database:
  - For OCP 3.11, pull the Fuse Online backup and restore database images from docker.io to a local cache.
  - For OCP 4.6, add Docker Hub credentials into the existing syndesis pull secret.
- Download the latest Fuse Online release.
- Obtain permission to upgrade Fuse Online from a cluster administrator.

- Run the update script.

The upgrade procedure for the following upgrades is the same:

- From Fuse Online 7.7 to Fuse Online 7.8
- From a Fuse Online 7.8 version to a newer Fuse Online 7.8 version

Prerequisites

- You installed and are running version 7.7 of Fuse Online on OCP on-site. **OR**, you installed and are running a version of 7.8 of Fuse Online on OCP 3.11 and you want to upgrade to fresh application images.
  For earlier versions:
  - If you are running version 7.6 of Fuse Online on OCP, then you must upgrade to 7.7 and then you can upgrade to 7.8.
  - If you are running version 7.5 of Fuse Online on OCP, then you must upgrade to 7.6 and then you can upgrade to 7.7.
  - If you are running version 7.4 of Fuse Online on OCP, then you must upgrade to 7.5 and then you can upgrade to 7.6.
  - If you are running version 7.3 of Fuse Online on OCP, then you must upgrade to 7.4 and then you can upgrade to 7.5.
  - If you are running version 7.2 of Fuse Online on OCP, then you must upgrade to 7.3.
  - If you are running version 7.1 of Fuse Online on OCP, then you must upgrade to 7.2.
- You installed the **oc** client tool and it is connected to the OCP cluster in which Fuse Online is installed.
- You have cluster administration permissions, which are required for the first two steps in this procedure.

Procedure

1. To avoid encountering a possible Docker limit error, a cluster administrator sets up access to Docker images as described in Access Docker images before an upgrade .
2. A cluster administrator downloads the Fuse Online package and grants permission for a user to upgrade Fuse Online in a particular project:
   a. Download the package containing the Fuse Online installation scripts from the following location:
   https://github.com/syndesisio/fuse-online-install/releases/tag/1.11

   Unpack the downloaded archive at a convenient location on your file system. The **fuse-online-install-1.11** directory contains the scripts and supporting files for upgrading Fuse Online.

   b. Change to the directory that contains the extracted archive. For example:
   cd fuse-online-install-1.11

   c. Log in to OpenShift with a cluster administration account, for example:

oc login -u admin -p admin

d. Change to the OpenShift project in which Fuse Online needs to be upgraded, for example:
oc project fuse-online-project

e. Update the Fuse Online custom resource definition:
bash install_ocp.sh --setup

f. Grant permission for upgrading Fuse Online in just this project. For example, the following command grants permission for upgrading Fuse Online to the **developer** user. After the cluster administrator runs this command, the **developer** user can upgrade Fuse Online in only this project, which is **fuse-online-project**, in this example:
bash install_ocp.sh --grant developer

3. The user who was granted permission to upgrade Fuse Online performs the upgrade:

a. Log in to OpenShift, for example:
oc login -u developer

b. Switch to the project in which you want to upgrade Fuse Online, for example:
oc project fuse-online-project

c. To check which version you are about to upgrade to, run the update script with the **--version** option, as follows:
bash update_ocp.sh --version

d. Invoke the update script as follows:
bash update_ocp.sh

To learn more about the script, invoke **bash update_ocp.sh --help**.

During and after an infrastructure upgrade, existing integrations continue to run with the *older* versions of Fuse Online libraries and dependencies.

4. Upgrade Fuse Online integrations that are running as follows:

a. In Fuse Online, select the integration that you want to upgrade.

b. Select **Edit**.

c. Select **Publish** to republish the integration.

Republishing the integration forces a rebuild that uses the latest Fuse Online dependencies.

1.4. CAMEL MIGRATION CONSIDERATIONS

Red Hat Fuse uses Apache Camel 2.23. You should consider the following updates to Camel 2.22 and 2.23 when you upgrade to Fuse 7.8.

Camel 2.22 updates

- Camel has upgraded from Spring Boot v1 to v2 and therefore v1 is no longer supported.
- Upgraded to Spring Framework 5. Camel should work with Spring 4.3.x as well, but going forward Spring 5.x will be the minimum Spring version in future releases.

- Upgraded to Karaf 4.2. You may run Camel on Karaf 4.1 but we only officially support Karaf 4.2 in this release.
- Optimized using toD DSL to reuse endpoints and producers for components where it is possible. For example, HTTP based components will now reuse producer (HTTP clients) with dynamic URIs sending to the same host.
- The File2 consumer with read-lock idempotent/idempotent-changed can now be configured to delay the release tasks to expand the window when a file is regarded as in-process, which is usable in active/active cluster settings with a shared idempotent repository to ensure other nodes don't too quickly see a processed file as a file they can process (only needed if you have readLockRemoveOnCommit=true).
- Allow to plugin a custom request/reply correlation id manager implementation on Netty4 producer in request/reply mode. The Twitter component now uses extended mode by default to support tweets greater than 140 characters
- Rest DSL producer now supports being configured in REST configuration by using endpointProperties.
- The Kafka component now supports HeaderFilterStrategy to plugin custom implementations for controlling header mappings between Camel and Kafka messages.
- REST DSL now supports client request validation to validate that Content-Type/Accept headers are possible for the REST service.
- Camel now has a Service Registry SPI which allows you to register routes to a service registry (such as consul, etcd, or zookeeper) by using a Camel implementation or Spring Cloud.
- The SEDA component now has a default queue size of 1000 instead of unlimited.
- The following noteworthy issues have been fixed:
  - Fixed a CXF continuation timeout issue with camel-cxf consumer that could cause the consumer to return a response with data instead of triggering a timeout to the calling SOAP client.
  - Fixed camel-cxf consumer doesn't release UoW when using a robust one-way operation.
  - Fixed using AdviceWith and using weave methods on onException etc. not working.
  - Fixed Splitter in parallel processing and streaming mode may block, while iterating message body when the iterator throws an exception in the first invoked next() method call.
  - Fixed Kafka consumer to not auto commit if autoCommitEnable=false.
  - Fixed file consumer was using markerFile as read-lock by default, which should have been none.
  - Fixed using manual commit with Kafka to provide the current record offset and not the previous (and -1 for first).
  - Fixed Content Based Router in Java DSL may not resolve property placeholders in when predicates.

Camel 2.23 updates

- Upgraded to Spring Boot 2.1.
- Additional component-level options can now be configured by using spring-boot auto-configuration. These options are included in the spring-boot component metadata JSON file descriptor for tooling assistance.
- Added a documentation section that includes all the Spring Boot auto configuration options for all the components, data-formats, and languages.
- All the Camel Spring Boot starter JARs now include **META-INF/spring-autoconfigure-metadata.properties** file in their JARs to optimize Spring Boot auto-configuration.
- The Throttler now supports correlation groups based on dynamic expression so that you can group messages into different throttled sets.
- The Hystrix EIP now allows inheritance for Camel's error handler so that you can retry the entire Hystrix EIP block again if you have enabled error handling with redeliveries.
- SQL and ElSql consumers now support dynamic query parameters in route form. Note that this feature is limited to calling beans by using simple expressions.
- The swagger-restdsl maven plugin now supports generating DTO model classes from the Swagger specification file.
- The following noteworthy issues have been fixed:
    - The Aggregator2 has been fixed to not propagate control headers for forcing completion of all groups, so it will not happen again if another aggregator EIP is in use later during routing.
    - Fixed Tracer not working if redelivery was activated in the error handler.
    - The built-in type converter for XML Documents may output parsing errors to stdout, which has now been fixed to output by using the logging API.
    - Fixed SFTP writing files by using the charset option would not work if the message body was streaming-based.
    - Fixed Zipkin root id to not be reused when routing over multiple routes to group them together into a single parent span.
    - Fixed optimized toD when using HTTP endpoints had a bug when the hostname contains an IP address with digits.
    - Fixed issue with RabbitMQ with request/reply over temporary queues and using manual acknowledge mode. It would not acknowledge the temporary queue (which is needed to make request/reply possible).
    - Fixed various HTTP consumer components that may not return all allowed HTTP verbs in Allow header for OPTIONS requests (such as when using rest-dsl).
    - Fixed the thread-safety issue with FluentProducerTemplate.

CHAPTER 2. UPGRADE TO SPRING BOOT 2

This chapter explains how to upgrade your applications to Spring Boot 2.0 from Spring Boot 1.

2.1. BEFORE YOU BEGIN

Before you start the migration to Spring Boot 2, you must review the system requirements and dependencies.

- Upgrade to the latest 1.5.x version
  - Before you start, upgrade to the latest 1.5.x available version. This is to ensure that you are building against the most recent dependencies of that line.
- Review dependencies
  - The migration to Spring Boot 2 will result in upgrading a number of dependencies. Review the dependency management for 1.5.x with the dependency management for 2.0.x to assess how your project is affected.
  - Identify the compatible versions for the dependencies that are not managed by Spring Boot and then define the explicit versions for these.
- Review custom configuration
  - Any custom configuration that your project defines might need to be reviewed on upgrade. If this can be replaced by the use of standard auto-configuration, do it so before upgrading.
- Review system requirements
  - Spring Boot 2.0 requires Java 8 or later.
  - It also requires Spring Framework 5.0.
  - Java 6 and 7 are no longer supported.

2.2. UPGRADE FROM SPRING BOOT 1 TO SPRING BOOT 2

Once you have reviewed the state of your project and its dependencies, upgrade to the latest maintenance release of Spring Boot 2.x. It is recommended to upgrade in the phases. For example, first upgrade from Spring Boot 1.5 to Spring Boot 2.0 and then upgrade to 2.1 and then to the latest maintenance release of Spring Boot 2.

Migrating configuration properties

With Spring Boot 2.0, many configuration properties were renamed or removed. Hence you need to update the **application.properties/application.yml** accordingly. You can achieve that with the help of a new **spring-boot-properties-migrator** module. Once added as a dependency to your project, this will not only analyze your application's environment and print diagnostics at startup, but also temporarily migrate properties at runtime for you.

Procedure

1. Add **spring-boot-properties-migrator** module to dependency section of your project's **pom.xml**.

```
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-properties-migrator</artifactId>
 <scope>runtime</scope>
</dependency>

runtime("org.springframework.boot:spring-boot-properties-migrator")
```

NOTE

Once you're done with the migration, please make sure to remove this module from your project's dependencies.

CHAPTER 3. UPGRADING FUSE APPLICATIONS ON SPRING BOOT STANDALONE

To upgrade your Fuse applications on Spring Boot:

- You should consider Apache Camel updates as described in Section 3.1, "Camel migration considerations".
- You must update your Fuse project's Maven dependencies to ensure that you are using the correct version of Fuse.

Typically, you use Maven to build Fuse applications. Maven is a free and open source build tool from Apache. Maven configuration is defined in a Fuse application project's **pom.xml** file. While building a Fuse project, the default behavior is that Maven searches external repositories and downloads the required artifacts. You add a dependency for the Fuse Bill of Materials (BOM) to the **pom.xml** file so that the Maven build process picks up the correct set of Fuse supported artifacts.

The following sections provide information on Maven dependencies and how to update them in your Fuse projects.

- Section 3.2, "About Maven dependencies"
- Section 3.3, "Updating your Fuse project's Maven dependencies"

3.1. CAMEL MIGRATION CONSIDERATIONS

Red Hat Fuse uses Apache Camel 2.23. You should consider the following updates to Camel 2.22 and 2.23 when you upgrade to Fuse 7.8.

Camel 2.22 updates

- Camel has upgraded from Spring Boot v1 to v2 and therefore v1 is no longer supported.
- Upgraded to Spring Framework 5. Camel should work with Spring 4.3.x as well, but going forward Spring 5.x will be the minimum Spring version in future releases.
- Upgraded to Karaf 4.2. You may run Camel on Karaf 4.1 but we only officially support Karaf 4.2 in this release.
- Optimized using toD DSL to reuse endpoints and producers for components where it is possible. For example, HTTP based components will now reuse producer (HTTP clients) with dynamic URIs sending to the same host.
- The File2 consumer with read-lock idempotent/idempotent-changed can now be configured to delay the release tasks to expand the window when a file is regarded as in-process, which is usable in active/active cluster settings with a shared idempotent repository to ensure other nodes don't too quickly see a processed file as a file they can process (only needed if you have readLockRemoveOnCommit=true).
- Allow to plugin a custom request/reply correlation id manager implementation on Netty4 producer in request/reply mode. The Twitter component now uses extended mode by default to support tweets greater than 140 characters
- Rest DSL producer now supports being configured in REST configuration by using endpointProperties.

- The Kafka component now supports HeaderFilterStrategy to plugin custom implementations for controlling header mappings between Camel and Kafka messages.
- REST DSL now supports client request validation to validate that Content-Type/Accept headers are possible for the REST service.
- Camel now has a Service Registry SPI which allows you to register routes to a service registry (such as consul, etcd, or zookeeper) by using a Camel implementation or Spring Cloud.
- The SEDA component now has a default queue size of 1000 instead of unlimited.
- The following noteworthy issues have been fixed:
    - Fixed a CXF continuation timeout issue with camel-cxf consumer that could cause the consumer to return a response with data instead of triggering a timeout to the calling SOAP client.
    - Fixed camel-cxf consumer doesn't release UoW when using a robust one-way operation.
    - Fixed using AdviceWith and using weave methods on onException etc. not working.
    - Fixed Splitter in parallel processing and streaming mode may block, while iterating message body when the iterator throws an exception in the first invoked next() method call.
    - Fixed Kafka consumer to not auto commit if autoCommitEnable=false.
    - Fixed file consumer was using markerFile as read-lock by default, which should have been none.
    - Fixed using manual commit with Kafka to provide the current record offset and not the previous (and -1 for first).
    - Fixed Content Based Router in Java DSL may not resolve property placeholders in when predicates.

Camel 2.23 updates

- Upgraded to Spring Boot 2.1.
- Additional component-level options can now be configured by using spring-boot auto-configuration. These options are included in the spring-boot component metadata JSON file descriptor for tooling assistance.
- Added a documentation section that includes all the Spring Boot auto configuration options for all the components, data-formats, and languages.
- All the Camel Spring Boot starter JARs now include **META-INF/spring-autoconfigure-metadata.properties** file in their JARs to optimize Spring Boot auto-configuration.
- The Throttler now supports correlation groups based on dynamic expression so that you can group messages into different throttled sets.
- The Hystrix EIP now allows inheritance for Camel's error handler so that you can retry the entire Hystrix EIP block again if you have enabled error handling with redeliveries.
- SQL and ElSql consumers now support dynamic query parameters in route form. Note that this feature is limited to calling beans by using simple expressions.

- The swagger-restdsl maven plugin now supports generating DTO model classes from the Swagger specification file.
- The following noteworthy issues have been fixed:
  - The Aggregator2 has been fixed to not propagate control headers for forcing completion of all groups, so it will not happen again if another aggregator EIP is in use later during routing.
  - Fixed Tracer not working if redelivery was activated in the error handler.
  - The built-in type converter for XML Documents may output parsing errors to stdout, which has now been fixed to output by using the logging API.
  - Fixed SFTP writing files by using the charset option would not work if the message body was streaming-based.
  - Fixed Zipkin root id to not be reused when routing over multiple routes to group them together into a single parent span.
  - Fixed optimized toD when using HTTP endpoints had a bug when the hostname contains an IP address with digits.
  - Fixed issue with RabbitMQ with request/reply over temporary queues and using manual acknowledge mode. It would not acknowledge the temporary queue (which is needed to make request/reply possible).
  - Fixed various HTTP consumer components that may not return all allowed HTTP verbs in Allow header for OPTIONS requests (such as when using rest-dsl).
  - Fixed the thread-safety issue with FluentProducerTemplate.

3.2. ABOUT MAVEN DEPENDENCIES

The purpose of a Maven Bill of Materials (BOM) file is to provide a curated set of Maven dependency versions that work well together, saving you from having to define versions individually for every Maven artifact.

There is a dedicated BOM file for each container in which Fuse runs.

NOTE

You can find these BOM files here: https://github.com/jboss-fuse/redhat-fuse. Alternatively, go to the latest Release Notes for information on BOM file updates.

The Fuse BOM offers the following advantages:

- Defines versions for Maven dependencies, so that you do not need to specify the version when you add a dependency to your **pom.xml** file.
- Defines a set of curated dependencies that are fully tested and supported for a specific version of Fuse.
- Simplifies upgrades of Fuse.

IMPORTANT

Only the set of dependencies defined by a Fuse BOM are supported by Red Hat.

3.3. UPDATING YOUR FUSE PROJECT'S MAVEN DEPENDENCIES

To upgrade your Fuse application for Spring Boot, update your project's Maven dependencies.

Procedure

1. Open your project's **pom.xml** file.
2. Add a **dependencyManagement** element in your project's **pom.xml** file (or, possibly, in a parent **pom.xml** file), as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.8.0.fuse-sb2-780038-redhat-00001</fuse.version>

  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.redhat-fuse</groupId>
        <artifactId>fuse-springboot-bom</artifactId>
        <version>${fuse.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
</project>
```

NOTE

Ensure you update your Spring Boot version as well. This is typically found under the Fuse version in the **pom.xml** file:

```
  <properties>
    <!-- configure the versions you want to use here -->
    <fuse.version>7.8.0.fuse-sb2-780038-redhat-00001</fuse.version>
    <spring-boot.version>2.3.4.RELEASE</spring-boot.version>
  </properties>
```

3. Save your **pom.xml** file.

After you specify the BOM as a dependency in your **pom.xml** file, it becomes possible to add Maven

dependencies to your **pom.xml** file *without* specifying the version of the artifact. For example, to add a dependency for the **camel-velocity** component, you would add the following XML fragment to the **dependencies** element in your **pom.xml** file:

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <scope>provided</scope>
</dependency>
```

Note how the **version** element is omitted from this dependency definition.

CHAPTER 4. UPGRADING FUSE APPLICATIONS ON JBOSS EAP STANDALONE

To upgrade your Fuse applications on JBoss EAP:

- You should consider Apache Camel updates as described in Section 4.1, "Camel migration considerations".
- You must update your Fuse project's Maven dependencies to ensure that you are using the correct version of Fuse.
  Typically, you use Maven to build Fuse applications. Maven is a free and open source build tool from Apache. Maven configuration is defined in a Fuse application project's **pom.xml** file. While building a Fuse project, the default behavior is that Maven searches external repositories and downloads the required artifacts. You add a dependency for the Fuse Bill of Materials (BOM) to the **pom.xml** file so that the Maven build process picks up the correct set of Fuse supported artifacts.

  The following sections provide information on Maven dependencies and how to update them in your Fuse projects.

  - Section 4.2, "About Maven dependencies"
  - Section 4.3, "Updating your Fuse project's Maven dependencies"
- You must update your Fuse project's Maven dependencies to ensure that you are using the upgraded versions of the Java EE dependencies as described in Section 4.4, "Upgrading your Java EE dependencies".

4.1. CAMEL MIGRATION CONSIDERATIONS

Red Hat Fuse uses Apache Camel 2.23. You should consider the following updates to Camel 2.22 and 2.23 when you upgrade to Fuse 7.8.

Camel 2.22 updates

- Camel has upgraded from Spring Boot v1 to v2 and therefore v1 is no longer supported.
- Upgraded to Spring Framework 5. Camel should work with Spring 4.3.x as well, but going forward Spring 5.x will be the minimum Spring version in future releases.
- Upgraded to Karaf 4.2. You may run Camel on Karaf 4.1 but we only officially support Karaf 4.2 in this release.
- Optimized using toD DSL to reuse endpoints and producers for components where it is possible. For example, HTTP based components will now reuse producer (HTTP clients) with dynamic URIs sending to the same host.
- The File2 consumer with read-lock idempotent/idempotent-changed can now be configured to delay the release tasks to expand the window when a file is regarded as in-process, which is usable in active/active cluster settings with a shared idempotent repository to ensure other nodes don't too quickly see a processed file as a file they can process (only needed if you have readLockRemoveOnCommit=true).

- Allow to plugin a custom request/reply correlation id manager implementation on Netty4 producer in request/reply mode. The Twitter component now uses extended mode by default to support tweets greater than 140 characters
- Rest DSL producer now supports being configured in REST configuration by using endpointProperties.
- The Kafka component now supports HeaderFilterStrategy to plugin custom implementations for controlling header mappings between Camel and Kafka messages.
- REST DSL now supports client request validation to validate that Content-Type/Accept headers are possible for the REST service.
- Camel now has a Service Registry SPI which allows you to register routes to a service registry (such as consul, etcd, or zookeeper) by using a Camel implementation or Spring Cloud.
- The SEDA component now has a default queue size of 1000 instead of unlimited.
- The following noteworthy issues have been fixed:
  - Fixed a CXF continuation timeout issue with camel-cxf consumer that could cause the consumer to return a response with data instead of triggering a timeout to the calling SOAP client.
  - Fixed camel-cxf consumer doesn't release UoW when using a robust one-way operation.
  - Fixed using AdviceWith and using weave methods on onException etc. not working.
  - Fixed Splitter in parallel processing and streaming mode may block, while iterating message body when the iterator throws an exception in the first invoked next() method call.
  - Fixed Kafka consumer to not auto commit if autoCommitEnable=false.
  - Fixed file consumer was using markerFile as read-lock by default, which should have been none.
  - Fixed using manual commit with Kafka to provide the current record offset and not the previous (and -1 for first).
  - Fixed Content Based Router in Java DSL may not resolve property placeholders in when predicates.

Camel 2.23 updates

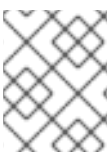
- Upgraded to Spring Boot 2.1.
- Additional component-level options can now be configured by using spring-boot auto-configuration. These options are included in the spring-boot component metadata JSON file descriptor for tooling assistance.
- Added a documentation section that includes all the Spring Boot auto configuration options for all the components, data-formats, and languages.
- All the Camel Spring Boot starter JARs now include **META-INF/spring-autoconfigure-metadata.properties** file in their JARs to optimize Spring Boot auto-configuration.

- The Throttler now supports correlation groups based on dynamic expression so that you can group messages into different throttled sets.
- The Hystrix EIP now allows inheritance for Camel's error handler so that you can retry the entire Hystrix EIP block again if you have enabled error handling with redeliveries.
- SQL and ElSql consumers now support dynamic query parameters in route form. Note that this feature is limited to calling beans by using simple expressions.
- The swagger-restdsl maven plugin now supports generating DTO model classes from the Swagger specification file.
- The following noteworthy issues have been fixed:
  - The Aggregator2 has been fixed to not propagate control headers for forcing completion of all groups, so it will not happen again if another aggregator EIP is in use later during routing.
  - Fixed Tracer not working if redelivery was activated in the error handler.
  - The built-in type converter for XML Documents may output parsing errors to stdout, which has now been fixed to output by using the logging API.
  - Fixed SFTP writing files by using the charset option would not work if the message body was streaming-based.
  - Fixed Zipkin root id to not be reused when routing over multiple routes to group them together into a single parent span.
  - Fixed optimized toD when using HTTP endpoints had a bug when the hostname contains an IP address with digits.
  - Fixed issue with RabbitMQ with request/reply over temporary queues and using manual acknowledge mode. It would not acknowledge the temporary queue (which is needed to make request/reply possible).
  - Fixed various HTTP consumer components that may not return all allowed HTTP verbs in Allow header for OPTIONS requests (such as when using rest-dsl).
  - Fixed the thread-safety issue with FluentProducerTemplate.

4.2. ABOUT MAVEN DEPENDENCIES

The purpose of a Maven Bill of Materials (BOM) file is to provide a curated set of Maven dependency versions that work well together, saving you from having to define versions individually for every Maven artifact.

There is a dedicated BOM file for each container in which Fuse runs.



NOTE

You can find these BOM files here: https://github.com/jboss-fuse/redhat-fuse. Alternatively, go to the latest Release Notes for information on BOM file updates.

The Fuse BOM offers the following advantages:

- Defines versions for Maven dependencies, so that you do not need to specify the version when you add a dependency to your **pom.xml** file.
- Defines a set of curated dependencies that are fully tested and supported for a specific version of Fuse.
- Simplifies upgrades of Fuse.

IMPORTANT

Only the set of dependencies defined by a Fuse BOM are supported by Red Hat.

4.3. UPDATING YOUR FUSE PROJECT'S MAVEN DEPENDENCIES

To upgrade your Fuse application for JBoss EAP, update your project's Maven dependencies.

Procedure

1. Open your project's **pom.xml** file.
2. Add a **dependencyManagement** element in your project's **pom.xml** file (or, possibly, in a parent **pom.xml** file), as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.8.0.fuse-sb2-780038-redhat-00001</fuse.version>

  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.redhat-fuse</groupId>
        <artifactId>fuse-eap-bom</artifactId>
        <version>${fuse.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
</project>
```

3. Save your **pom.xml** file.

After you specify the BOM as a dependency in your **pom.xml** file, it becomes possible to add Maven dependencies to your **pom.xml** file *without* specifying the version of the artifact. For example, to add a dependency for the **camel-velocity** component, you would add the following XML fragment to the **dependencies** element in your **pom.xml** file:

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <scope>provided</scope>
</dependency>
```

Note how the **version** element is omitted from this dependency definition.

4.4. UPGRADING YOUR JAVA EE DEPENDENCIES

In Fuse 7.8, some managed dependencies in the BOM file have updated **groupId** or **artifactId** properties, therefore you must update your project's **pom.xml** file accordingly.

Procedure

1. Open your project's **pom.xml** file.
2. To change the **org.jboss.spec.javax.transaction** version from 1.2 to 1.3 and the **org.jboss.spec.javax.servlet** version from 3.1 to 4.0, update the dependencies as shown in the following example:

```
<dependency>
  <groupId>org.jboss.spec.javax.transaction</groupId>
  <artifactId>jboss-transaction-api_1.3_spec</artifactId>
</dependency>

<dependency>
    <groupId>org.jboss.spec.javax.servlet</groupId>
    <artifactId>jboss-servlet-api_4.0_spec</artifactId>
</dependency>
```

3. To migrate from Java EE API to Jakarta EE, replace **javax.*** with **jakarta.*** for each **groupId** and modify the **artifactId** for individual dependencies as shown in the following example:

```
<dependency>
  <groupId>jakarta.validation</groupId>
  <artifactId>jakarta.validation-api</artifactId>
</dependency>

<dependency>
  <groupId>jakarta.enterprise</groupId>
  <artifactId>jakarta.enterprise.cdi-api</artifactId>
</dependency>

<dependency>
  <groupId>jakarta.inject</groupId>
  <artifactId>jakarta.inject-api</artifactId>
</dependency>
```

4.5. UPGRADING AN EXISTING FUSE ON JBOSS EAP INSTALLATION

The following procedure describes how to upgrade an existing Fuse on JBoss EAP installation.

Procedure

1. To upgrade from one JBoss EAP minor release to another, you should follow the instructions in the JBoss EAP Patching and Upgrading Guide guide.

2. To update Fuse, you must run the Fuse on JBoss EAP installer as described in the Installing on JBoss EAP guide.

NOTE

You should not need to recompile or redploy your Fuse application.

4.6. UPGRADING FUSE AND JBOSS EAP SIMULTANEOUSLY

The following procedure describes how to upgrade a Fuse installation and the JBoss EAP runtime simultanously, for example, if you are migrating from Fuse 7.7 on JBoss EAP 7.2 to Fuse 7.8 on JBoss EAP 7.3.

WARNING

When upgrading **both** Fuse and the JBoss EAP runtime, Red Hat recommends that you perform a fresh installation of both Fuse and the JBoss EAP runtime.

Procedure

1. To perform a new installation of the JBoss EAP runtime, follow the instructions in the Installing on JBoss EAP guide.

2. To perform a new installation of Fuse, run the Fuse on JBoss EAP installer as described in the Installing on JBoss EAP guide.

CHAPTER 5. UPGRADING FUSE APPLICATIONS ON KARAF STANDALONE

To upgrade your Fuse applications on Karaf:

- You should consider Apache Camel updates as described in Section 5.1, “Camel migration considerations”.
- You must update your Fuse project’s Maven dependencies to ensure that you are using the correct version of Fuse.

Typically, you use Maven to build Fuse applications. Maven is a free and open source build tool from Apache. Maven configuration is defined in a Fuse application project’s **pom.xml** file. While building a Fuse project, the default behavior is that Maven searches external repositories and downloads the required artifacts. You add a dependency for the Fuse Bill of Materials (BOM) to the **pom.xml** file so that the Maven build process picks up the correct set of Fuse supported artifacts.

The following sections provide information on Maven dependencies and how to update them in your Fuse projects.

- Section 5.2, “About Maven dependencies”
- Section 5.3, “Updating your Fuse project’s Maven dependencies”

5.1. CAMEL MIGRATION CONSIDERATIONS

Red Hat Fuse uses Apache Camel 2.23. You should consider the following updates to Camel 2.22 and 2.23 when you upgrade to Fuse 7.8.

Camel 2.22 updates

- Camel has upgraded from Spring Boot v1 to v2 and therefore v1 is no longer supported.
- Upgraded to Spring Framework 5. Camel should work with Spring 4.3.x as well, but going forward Spring 5.x will be the minimum Spring version in future releases.
- Upgraded to Karaf 4.2. You may run Camel on Karaf 4.1 but we only officially support Karaf 4.2 in this release.
- Optimized using toD DSL to reuse endpoints and producers for components where it is possible. For example, HTTP based components will now reuse producer (HTTP clients) with dynamic URIs sending to the same host.
- The File2 consumer with read-lock idempotent/idempotent-changed can now be configured to delay the release tasks to expand the window when a file is regarded as in-process, which is usable in active/active cluster settings with a shared idempotent repository to ensure other nodes don’t too quickly see a processed file as a file they can process (only needed if you have readLockRemoveOnCommit=true).
- Allow to plugin a custom request/reply correlation id manager implementation on Netty4 producer in request/reply mode. The Twitter component now uses extended mode by default to support tweets greater than 140 characters
- Rest DSL producer now supports being configured in REST configuration by using endpointProperties.

- The Kafka component now supports HeaderFilterStrategy to plugin custom implementations for controlling header mappings between Camel and Kafka messages.
- REST DSL now supports client request validation to validate that Content-Type/Accept headers are possible for the REST service.
- Camel now has a Service Registry SPI which allows you to register routes to a service registry (such as consul, etcd, or zookeeper) by using a Camel implementation or Spring Cloud.
- The SEDA component now has a default queue size of 1000 instead of unlimited.
- The following noteworthy issues have been fixed:
  - Fixed a CXF continuation timeout issue with camel-cxf consumer that could cause the consumer to return a response with data instead of triggering a timeout to the calling SOAP client.
  - Fixed camel-cxf consumer doesn't release UoW when using a robust one-way operation.
  - Fixed using AdviceWith and using weave methods on onException etc. not working.
  - Fixed Splitter in parallel processing and streaming mode may block, while iterating message body when the iterator throws an exception in the first invoked next() method call.
  - Fixed Kafka consumer to not auto commit if autoCommitEnable=false.
  - Fixed file consumer was using markerFile as read-lock by default, which should have been none.
  - Fixed using manual commit with Kafka to provide the current record offset and not the previous (and -1 for first).
  - Fixed Content Based Router in Java DSL may not resolve property placeholders in when predicates.

Camel 2.23 updates

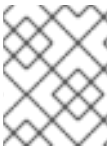
- Upgraded to Spring Boot 2.1.
- Additional component-level options can now be configured by using spring-boot auto-configuration. These options are included in the spring-boot component metadata JSON file descriptor for tooling assistance.
- Added a documentation section that includes all the Spring Boot auto configuration options for all the components, data-formats, and languages.
- All the Camel Spring Boot starter JARs now include **META-INF/spring-autoconfigure-metadata.properties** file in their JARs to optimize Spring Boot auto-configuration.
- The Throttler now supports correlation groups based on dynamic expression so that you can group messages into different throttled sets.
- The Hystrix EIP now allows inheritance for Camel's error handler so that you can retry the entire Hystrix EIP block again if you have enabled error handling with redeliveries.
- SQL and ElSql consumers now support dynamic query parameters in route form. Note that this feature is limited to calling beans by using simple expressions.

- The swagger-restdsl maven plugin now supports generating DTO model classes from the Swagger specification file.
- The following noteworthy issues have been fixed:
  - The Aggregator2 has been fixed to not propagate control headers for forcing completion of all groups, so it will not happen again if another aggregator EIP is in use later during routing.
  - Fixed Tracer not working if redelivery was activated in the error handler.
  - The built-in type converter for XML Documents may output parsing errors to stdout, which has now been fixed to output by using the logging API.
  - Fixed SFTP writing files by using the charset option would not work if the message body was streaming-based.
  - Fixed Zipkin root id to not be reused when routing over multiple routes to group them together into a single parent span.
  - Fixed optimized toD when using HTTP endpoints had a bug when the hostname contains an IP address with digits.
  - Fixed issue with RabbitMQ with request/reply over temporary queues and using manual acknowledge mode. It would not acknowledge the temporary queue (which is needed to make request/reply possible).
  - Fixed various HTTP consumer components that may not return all allowed HTTP verbs in Allow header for OPTIONS requests (such as when using rest-dsl).
  - Fixed the thread-safety issue with FluentProducerTemplate.

5.2. ABOUT MAVEN DEPENDENCIES

The purpose of a Maven Bill of Materials (BOM) file is to provide a curated set of Maven dependency versions that work well together, saving you from having to define versions individually for every Maven artifact.

There is a dedicated BOM file for each container in which Fuse runs.

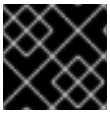


NOTE

You can find these BOM files here: https://github.com/jboss-fuse/redhat-fuse. Alternatively, go to the latest Release Notes for information on BOM file updates.

The Fuse BOM offers the following advantages:

- Defines versions for Maven dependencies, so that you do not need to specify the version when you add a dependency to your **pom.xml** file.
- Defines a set of curated dependencies that are fully tested and supported for a specific version of Fuse.
- Simplifies upgrades of Fuse.



IMPORTANT

Only the set of dependencies defined by a Fuse BOM are supported by Red Hat.

5.3. UPDATING YOUR FUSE PROJECT'S MAVEN DEPENDENCIES

To upgrade your Fuse application for Karaf, update your project's Maven dependencies.

Procedure

1. Open your project's **pom.xml** file.
2. Add a **dependencyManagement** element in your project's **pom.xml** file (or, possibly, in a parent **pom.xml** file), as shown in the following example:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project ...>
  ...
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <!-- configure the versions you want to use here -->
    <fuse.version>7.8.0.fuse-sb2-780038-redhat-00001</fuse.version>

  </properties>

  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>org.jboss.redhat-fuse</groupId>
        <artifactId>fuse-karaf-bom</artifactId>
        <version>${fuse.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
</project>
```

3. Save your **pom.xml** file.

After you specify the BOM as a dependency in your **pom.xml** file, it becomes possible to add Maven dependencies to your **pom.xml** file *without* specifying the version of the artifact. For example, to add a dependency for the **camel-velocity** component, you would add the following XML fragment to the **dependencies** element in your **pom.xml** file:

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-velocity</artifactId>
  <scope>provided</scope>
</dependency>
```

Note how the **version** element is omitted from this dependency definition.

CHAPTER 6. UPGRADING FUSE STANDALONE ON KARAF

In this release there are a number of upgrades which affect major and minor component versions. Most OSGi bundles set version ranges that exclude the next major version or sometimes even a minor version.

WARNING

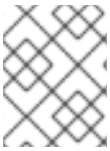
Do not use the Fuse on Apache Karaf patch mechanism to upgrade an Apache Karaf container to Fuse 7.8, a new installation must be performed, with configuration and other modified files copied across manually.

6.1. UPGRADING FUSE STANDALONE ON KARAF

The following instructions guide you through upgrading to Fuse 7.8 on Apache Karaf.

Procedure

1. To install Fuse on Apache Karaf, follow the procedure in Installing on Apache Karaf.
2. Rebuild the applications using new Fuse Karaf BOM for Fuse 7.8 (see the latest Release Notes for information on BOM file updates).
3. Install the features that were previously installed in Fuse 7.7
4. Compare the **etc** directories for potential configuration changes, for example, **logging**, **web** or **Maven**.
5. Compare **bin** directories for potential environment changes, for example, **bin/setenv**.
6. Install the rebuilt applications into Fuse 7.8



NOTE

After upgrading, you will see the new version and build number in the Welcome banner when you restart the container.