Red Hat Fuse 7.7

Migration Guide

Migrating to Red Hat Fuse 7.7

Last Updated: 2023-07-24

Red Hat Fuse 7.7 Migration Guide

Migrating to Red Hat Fuse 7.7

Legal Notice

Abstract

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

Table of Contents

PREFACE 3

CHAPTER 1. UPGRADING FUSE ONLINE 4
1.1. UPGRADING FUSE ONLINE INTEGRATIONS THAT ARE RUNNING ON OPENSHIFT ONLINE 4
1.2. UPGRADING FUSE ONLINE ON OCP 4

CHAPTER 2. UPGRADING FUSE APPLICATIONS ON SPRING BOOT STANDALONE 7
2.1. ABOUT MAVEN DEPENDENCIES 7
2.2. UPDATING YOUR FUSE PROJECT'S MAVEN DEPENDENCIES 7

CHAPTER 3. UPGRADING FUSE APPLICATIONS ON JBOSS EAP STANDALONE 9
3.1. ABOUT MAVEN DEPENDENCIES 9
3.2. UPDATING YOUR FUSE PROJECT'S MAVEN DEPENDENCIES 9
3.3. UPGRADING AN EXISTING FUSE ON JBOSS EAP INSTALLATION 10

CHAPTER 4. UPGRADING FUSE APPLICATIONS ON KARAF STANDALONE 12
4.1. ABOUT MAVEN DEPENDENCIES 12
4.2. UPDATING YOUR FUSE PROJECT'S MAVEN DEPENDENCIES 12

CHAPTER 5. UPGRADING FUSE STANDALONE ON KARAF 14
5.1. IMPACT OF UPGRADING FUSE ON KARAF 14
5.2. UPGRADING FUSE STANDALONE ON KARAF 14
5.3. ROLLING BACK AN UPGRADE FOR FUSE ON KARAF 16

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*

Chapter 2, *Upgrading Fuse applications on Spring Boot standalone*

Chapter 3, *Upgrading Fuse applications on JBoss EAP standalone*

Chapter 4, *Upgrading Fuse applications on Karaf standalone*

Chapter 5, *Upgrading Fuse Standalone on Karaf*

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 7.7 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 must download the latest Fuse Online release, run the update script, and then republish any running integrations as described in Upgrading Fuse Online on OCP .

1.1. UPGRADING FUSE ONLINE INTEGRATIONS THAT ARE RUNNING ON OPENSHIFT ONLINE

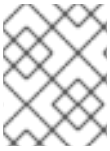
When Fuse 7.7 is released, the Fuse Online infrastructure on OpenShift Online is automatically upgraded. During the infrastructure upgrade, any existing integrations that are running on OpenShift Online continue to run both during and after the upgrade. However, the existing integrations continue to run with the *older* versions of Fuse libraries and dependencies.

After you receive an email message that lets you know that the Fuse Online infrastructure has been upgraded to the new release, upgrade your existing integrations by republishing them (not just restarting them). Do this as soon as you can.

To republish your integrations, in your Fuse Online environment, in the left navigation panel, click **Integrations**. Then do the following for each integration:

1. To the right of the integration entry, click  and select **Edit**.
2. When Fuse Online displays the integration for editing, in the upper right, click **Publish**.

Publishing forces a rebuild that uses the latest Fuse Online dependencies.



NOTE

The Fuse Online user interface shows a warning if any element of an integration has a newer dependency that needs to be updated.

1.2. UPGRADING FUSE ONLINE ON OCP

To upgrade Fuse Online on OCP on-site, download the latest Fuse Online release, obtain permission to upgrade Fuse Online from a cluster administrator, and run the update script.

From time to time, fresh application images, which incorporate patches and security fixes, are released for Fuse Online. You are notified of these updates through Red Hat's errata update channel. You can then upgrade your Fuse Online images.

The upgrade procedure for the following upgrades is the same:

- From Fuse Online 7.6 to Fuse Online 7.7
- From a Fuse Online 7.7 version to a newer Fuse Online 7.7 version

Prerequisites

- You installed and are running version 7.6 of Fuse Online on OCP on-site. **OR**, you installed and are running a version of 7.7 of Fuse Online on OCP on-site and you want to upgrade to fresh application images.
  For earlier versions:
  - 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 five steps in this procedure.

Procedure

1. 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/1.14

   Unpack the downloaded archive at a convenient location on your file system. The **fuse-online-install-1.14** 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-1.14

   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. 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
2. 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.

3. 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.

CHAPTER 2. UPGRADING FUSE APPLICATIONS ON SPRING BOOT STANDALONE

To upgrade your Fuse applications on Spring Boot, 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.

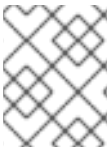
The following sections provide information on Maven dependencies and how to update them in your Fuse projects.

- Section 2.1, "About Maven dependencies"
- Section 2.2, "Updating your Fuse project's Maven dependencies"

2.1. 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.

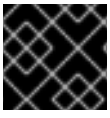


NOTE

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.

2.2. 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>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</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>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</fuse.version>
    <spring-boot.version>2.5.13.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 3. UPGRADING FUSE APPLICATIONS ON JBOSS EAP STANDALONE

To upgrade your Fuse applications on JBoss EAP, 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 3.1, "About Maven dependencies"
- Section 3.2, "Updating your Fuse project's Maven dependencies"

3.1. 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.

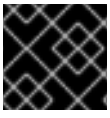
There is a dedicated BOM file for each container in which Fuse runs.

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.

3.2. 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>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</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.

3.3. 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.

CHAPTER 4. UPGRADING FUSE APPLICATIONS ON KARAF STANDALONE

To upgrade your Fuse applications on Karaf, 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.

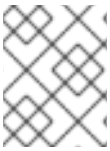
The following sections provide information on Maven dependencies and how to update them in your Fuse projects.

- Section 4.1, "About Maven dependencies"
- Section 4.2, "Updating your Fuse project's Maven dependencies"

4.1. 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.

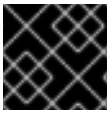


NOTE

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.

4.2. 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>7.11.1.fuse-sb2-7_11_1-00022-redhat-00002</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 5. UPGRADING FUSE STANDALONE ON KARAF

The Fuse on Apache Karaf upgrade mechanism enables you to apply fixes to an Apache Karaf container without needing to reinstall an updated version of Fuse on Karaf. It also allows you to roll back the upgrade, if the upgrade causes problems with your deployed applications.

The upgrade installer file is the *same file* that you use to install Fuse on Apache Karaf.

NOTE

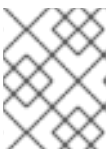
To obtain the upgrade installer file, go to the **Downloads** page of the Red Hat customer portal and download the latest version of the installation archive for Fuse on Apache Karaf (for example, **fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003.zip**).

- Section 5.1, “Impact of upgrading Fuse on Karaf”
- Section 5.2, “Upgrading Fuse Standalone on Karaf”
- Section 5.3, “Rolling back an upgrade for Fuse on Karaf”

5.1. IMPACT OF UPGRADING FUSE ON KARAF

The upgrade mechanism can make updates to **any** installation files including **bundle JARs** and **static files** (including, for example, configuration files under the **etc/** directory). The Fuse on Apache Karaf upgrade process:

- Updates any files, including bundle JARs, configuration files, and any static files.
- Patches both the current container instance (and its runtime storage under the **data/** directory) and the underlying installation. Hence, patches are preserved after deleting a container instance.
- Updates all of the files related to Karaf features, including the features repository files and the features themselves. Hence, any features installed after the rollup patch will reference the correct patched dependencies.
- If necessary, updates configuration files (for example, files under **etc/**), automatically merging any configuration changes you have made with the configuration changes made by the patch. If merge conflicts occur, see the patch log for details of how they are handled.
- Most of the merge conflicts are resolved automatically. For example, the patch mechanism detects conflicts at property level for the property files. It detects whether it was a user or patch that changed any property. The change is preserved, if only one side changed the property.
- Tracks **all** of the changes made to the installation (including to static files), so that it is possible to roll back the patch.



NOTE

The rollup patching mechanism uses an internal git repository (located under **patches/.management/history**) to track the changes made.

5.2. UPGRADING FUSE STANDALONE ON KARAF

The following instructions guide you through upgrading Fuse on Apache Karaf. Ensure all prerequisites are completed before commencing the upgrade procedure.

Prerequisites

- Ensure you have a full backup of your Fuse on Apache Karaf installation before upgrading.
- Start the container, if it is not already running.

TIP

If the container is running in the background (or remotely), connect to the container using the SSH console client, **bin/client**.

- Add the upgrade installer file to the container's environment by invoking the **patch:add** command. For example, to add the **fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003.zip** upgrade installer file:

```
patch:add file:///path/to/fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003.zip
```

Procedure

1. Run the **patch:update** command. There is no need to restart the container.

```
karaf@root()> patch:update
Current patch mechanism version: 7.1.0.fuse-710023-redhat-00001
New patch mechanism version detected: 7.2.0.fuse-720035-redhat-00001
Uninstalling patch features in version 7.1.0.fuse-710023-redhat-00001
Installing patch features in version 7.2.0.fuse-720035-redhat-00001
```

2. Invoke the **patch:list** command to display a list of upgrade installers. In this list, the entries under the **[name]** heading are upgrade IDs. For example:

```
karaf@root()> patch:list
[name]                                  [installed] [rollup] [description]
fuse-karaf-7.2.0.fuse-720035-redhat-00001 false       true     fuse-karaf-7.2.0.fuse-720035-
redhat-00001
```

3. Simulate the upgrade by invoking the **patch:simulate** command and specifying the upgrade ID for the upgrade that you want to apply, as follows:

```
karaf@root()> patch:simulate fuse-karaf-7.2.0.fuse-720035-redhat-00001
INFO : org.jboss.fuse.modules.patch.patch-management (226): Installing rollup patch "fuse-
karaf-7.2.0.fuse-720035-redhat-00001"
========== Repositories to remove (9):
 - mvn:io.hawt/hawtio-karaf/2.0.0.fuse-710018-redhat-00002/xml/features
...
========== Repositories to add (9):
 - mvn:io.hawt/hawtio-karaf/2.0.0.fuse-720044-redhat-00001/xml/features
...
========== Repositories to keep (10):
 - mvn:org.apache.activemq/artemis-features/2.4.0.amq-711002-redhat-1/xml/features
...
```

```
========== Features to update (100):
[name]                  [version]                  [new version]
aries-blueprint           4.2.0.fuse-710024-redhat-00002    4.2.0.fuse-720061-redhat-00001
...
========== Bundles to update as part of features or core bundles (100):
[symbolic name]                                    [version]                  [new location]
io.hawt.hawtio-log                                 2.0.0.fuse-710018-redhat-00002
mvn:io.hawt/hawtio-log/2.0.0.fuse-720044-redhat-00001
...
========== Bundles to reinstall as part of features or core bundles (123):
[symbolic name]                                    [version]                  [location]
com.fasterxml.jackson.core.jackson-annotations             2.8.11
mvn:com.fasterxml.jackson.core/jackson-annotations/2.8.11
...
Simulation only - no files and runtime data will be modified.
karaf@root()>
```

This generates a log of the changes that will be made to the container when the upgrade is performed, but will not make any actual changes to the container. Review the simulation log to understand the changes that will be made to the container.

4. Upgrade the container by invoking the **patch:install** command and specifying the upgrade ID for the upgrade that you want to apply. For example:

```
karaf@root()> patch:install fuse-karaf-7.11.1.fuse-7_11_1-00013-redhat-00003
```

5. Validate the upgrade, by searching for one of the upgrade artifacts. For example, if you had just upgraded Fuse 7.1.0 to Fuse 7.2.0, you could search for bundles with the build number, 7_11_1-00017-redhat-00001, as follows:

```
karaf@root()> bundle:list -l | grep 7_11_1-00017-redhat-00001
 22 │ Active │  80 │ 7.11.1.fuse-7_11_1-00013-redhat-00003  │
mvn:org.jboss.fuse.modules/fuse-pax-transx-tm-narayana/7.11.1.fuse-7_11_1-00013-redhat-00003
188 │ Active │  80 │ 7.11.1.fuse-7_11_1-00013-redhat-00003  │
mvn:org.jboss.fuse.modules.patch/patch-commands/7.11.1.fuse-7_11_1-00013-redhat-00003
```

NOTE

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

5.3. ROLLING BACK AN UPGRADE FOR FUSE ON KARAF

Occasionally an upgrade might not work or might introduce new issues to a container. In these cases, you can easily roll back the upgrade and restore your system to its previous state using the **patch:rollback** command. This set of instructions guides you through this procedure.

Prerequisites

- You have recently upgraded Fuse on Karaf.
- You want to rollback the upgrade.

Procedure

1. Invoke the **patch:list** command to obtain the upgrade ID, **UPGRADE_ID**, of the most recently installed patch.

2. Invoke the **patch:rollback** command, as follows:

```
patch:rollback UPGRADE_ID
```

NOTE

In some cases the container needs to restart to roll back the upgrade. In these cases, the container restarts automatically. Due to the highly dynamic nature of the OSGi runtime, during the restart you might see some occasional errors related to incompatible classes. These errors are related to OSGi services that have just started or stopped and can be safely ignored.