Red Hat JBoss Fuse 6.2.1

Migration Guide

Migrating to Red Hat JBoss Fuse 6.2.1

Last Updated: 2017-09-21

Red Hat JBoss Fuse 6.2.1 Migration Guide

Migrating to Red Hat JBoss Fuse 6.2.1

JBoss A-MQ Docs Team
Content Services
fuse-docs-support@redhat.com

Legal Notice

Abstract

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

Table of Contents

CHAPTER 1. WHAT'S NEW

Abstract

This section describes the main features and changes in version 6.2.1.

1.1. NEW FEATURES

With the release of JBoss Fuse 6.2.1, you can now choose freely between two alternative container technologies: JEE (JBoss EAP), or OSGi (Apache Karaf). Both container types now offer a very similar technology stack (SwitchYard, Apache Camel, Apache CXF, and so on), so you can choose your preferred container type without limiting your options.

The main features in version 6.2.1 are:

- Completed merge of the two products, JBoss Fuse and JBoss Fuse Service Works, into the single new product, JBoss Fuse:
  - For next steps with the EAP/JEE container, see "Security on JBoss EAP".
  - For next steps with the Karaf/OSGi container, see "Installation on Apache Karaf".
- New patching mechanism, which is capable of patching any files in a JBoss Fuse installation, not just bundle JARs. With this new mechanism it is possible to eliminate manual steps in patch installation. See chapter "Applying Patches" in "Configuring and Running JBoss Fuse" for more details.
- Camel on EAP (Wildfly Camel) subsystem, for easier deployment of Apache Camel components on the JBoss EAP container. See chapter "Apache Camel on Red Hat JBoss EAP" in "Deploying into a Web Server" for more details.
- SwitchYard on Karaf is now fully supported. See chapter "Getting Started with SwitchYard on Apache Karaf" in "Installation on Apache Karaf" for more details.
- Transformation Tooling is now fully supported. See chapter "Getting Started with Data Transformation" in "Tooling User Guide" for more details.
- SAP Tooling is now fully supported. See chapter "Using the JBoss Fuse SAP Tool Suite" in "Tooling User Guide" for more details.
- Swagger is now fully supported. See section "Swagger Integration" in "Apache Camel Development Guide" for more details.
- The SwitchYard part of the Integration Package is now fully supported.

1.2. TECHNOLOGY PREVIEW FEATURES

This release of JBoss Fuse does not introduce any new technology preview features.

NOTE

For details on what *technical preview* means, see https://access.redhat.com/support/offerings/techpreview/.

1.3. IMPORTANT NOTES

Increased JAVA_MAX_MEM requirement

In JBoss Fuse 6.2.1, the recommended value for the maximum memory allocated to the JVM of the Apache Karaf container is 1024 MB. This value is specified in the ***InstallDir*/bin/setenv** file (or ***InstallDir*\bin\setenv.bat** on Windows O/S) in an Apache Karaf container installation.

For example, the default settings for the Apache Karaf JVM in the 6.2.1 distribution are now:

```
#
# The following section shows the possible configuration options for the
default
# karaf scripts
#
JAVA_MIN_MEM=512M # Minimum memory for the JVM
JAVA_MAX_MEM=1024M # Maximum memory for the JVM
JAVA_PERM_MEM=128M # Minimum perm memory for the JVM
JAVA_MAX_PERM_MEM=256M # Maximum memory for the JVM
...
```

Migration from Apache OpenJPA to Hibernate

Since JBoss Fuse 6.2.0, the Hibernate libraries have been shipped with JBoss Fuse. If you use the Java Persisence Architecture (JPA) in your applications, it is recommended that you migrate away from OpenJPA and adopt the Hibernate JPA implementation instead. Red Hat can offer better support on the Hibernate JPA implementation. Apache OpenJPA is now deprecated and support for OpenJPA is likely to be discontinued in a future release.

CHAPTER 2. DEPRECATED AND REMOVED FEATURES

BIN/DELETEFABRIC8 SCRIPT IS DEPRECATED

The **`bin/deletefabric8`** script is deprecated and will be removed in a future release.

BIN/PATCH SCRIPT IS DEPRECATED

The **`bin/patch`** script (**`bin\patch.bat`** on Windows O/S) is deprecated and will be removed in a future release.

BPEL IS DEPRECATED

BPEL (based on the Riftsaw project) is no longer being actively developed and will be removed from a future release of JBoss Fuse. If you are currently using BPEL, it is recommended that you consider migrating to the Red Hat JBoss BPM Suite (which is supported through the JBoss Fuse Integration Package).

S-RAMP IS DEPRECATED

The SOA Repository Artifact Model and Protocol (S-RAMP) component is deprecated in 6.2.1 and will be removed from a future release.

DESIGN TIME GOVERNANCE IS DEPRECATED

The Design Time Governance component is deprecated in 6.2.1 and not included with the product.

APACHE OPENJPA IS DEPRECATED

The Apache OpenJPA implementation of the Java Persistence API (JPA) is deprecated in 6.2.1. It is recommended that you use the Hibernate implementation instead.

SPRING DYNAMIC MODULES (SPRING-DM) IS DEPRECATED

Spring-DM (which integrates Spring XML with the OSGi service layer) is deprecated in 6.2.1 and you should use the Blueprint framework instead. Using Blueprint does not prevent you from using the Java libraries from the Spring framework: the latest version of Spring is compatible with Blueprint.

CHAPTER 3. MIGRATION PATHS FOR JBOSS FUSE 6.2.1

MIGRATION PATH FOR JBOSS FUSE 6.2.1 ON KARAF

The following version of JBoss Fuse can be migrated direct to JBoss Fuse 6.2.1 using the new patching process (see Chapter 4, *Migrating JBoss Fuse from 6.2.0 to 6.2.1 on Karaf*):

- Fuse 6.2.0

For older versions of JBoss Fuse (for example, 6.0 or 6.1), there is no automated migration path.

MIGRATION PATH FOR JBOSS FUSE 6.2.1 ON EAP

There is no automated migration path to JBoss Fuse 6.2.1 on EAP from JBoss Fuse Service Works 6.0 (FSW 6.0). Users of FSW 6.0, will need to make a new installation of JBoss Fuse 6.2.1 on JBoss EAP. After a successful installation, any existing deployments will need to be re-deployed to the new system. For installation information please see Installation on JBoss EAP and for deployment information see Deployment in the Management Console.

CHAPTER 4. MIGRATING JBOSS FUSE FROM 6.2.0 TO 6.2.1 ON KARAF

Abstract

If you have already deployed an instance of JBoss Fuse 6.2.0 Apache Karaf container, you can upgrade your 6.2.0 installation to version 6.2.1 using the new patching mechanism. The procedure for migrating a standalone container is different from the procedure for migrating a collection of Fabric containers. Follow the appropriate set of instructions for your system.

4.1. PATCHING A STANDALONE KARAF CONTAINER FROM 6.2.0 TO 6.2.1

Overview

The instructions in this section are for upgrading an existing JBoss Fuse 6.2.0 Apache Karaf installation, which is deployed as a standalone container (in other words, *not* using Fabric), to version 6.2.1.

New patching mechanism

Upgrading from JBoss Fuse 6.2.0 to 6.2.1 requires the new patching mechanism, which is implemented for the first time in 6.2.1. This presents a bootstrapping problem: the existing 6.2.0 installation must be enhanced to support the new patching mechanism before you can install patch version 6.2.1. The upgrade procedure therefore consists of two distinct phases, as follows:

1. Install the patch management enablement pack for 6.2.0, which replaces the existing patch mechanism with the new patching mechanism. In all other respects, the container remains at version 6.2.0.

2. Install the 6.2.1 patch in the standalone container using the new patch mechanism. After this step, the container is upgraded fully to version 6.2.1.

Initial system

The starting point for this patching procedures is assumed to be an installation of Red Hat JBoss Fuse 6.2.0 Full Install (**`jboss-fuse-full-6.2.0.redhat-133.zip`**). This can be a container instance that you have customized in various ways, by adding application bundles and features, or even by editing configuration files under the **`etc/`** directory.

NOTE

The new patching process is usually non-destructive, preserving any customizations made before the patch was applied. If a merge conflict cannot be resolved automatically, however, warning messages will be generated in the log file.

Download the required packages

To patch JBoss Fuse from 6.2.0 to 6.2.1 you require the following packages:

`patch-management-for-fuse-620-6.2.1.redhat-084.zip`

Available as the download file, Red Hat JBoss Fuse 6.2.1 on Karaf Update Installer.

jboss-fuse-full-6.2.1.redhat-084.zip

Available as the download file, Red Hat JBoss Fuse 6.2.1 on Karaf Installer.

Applying the 6.2.1 patch to a standalone container

To upgrade a JBoss Fuse 6.2.0 standalone container to version 6.2.1, proceed as follows:

1. Make a full backup of your JBoss Fuse 6.2.0 installation before attempting to apply the patch. In particular, if you made any custom changes to the **etc/org.ops4j.pax.logging.cfg** file, make sure that you back it up. The patch process for 6.2.1 will over-write this file and you will need to re-apply your changes to it.

2. Install the patch management enablement pack for 6.2.0, **patch-management-for-fuse-620-6.2.1.redhat-084.zip**, on top of your 6.2.0 installation. Use an archive utility to extract the contents on top of the existing 6.2.0 installation.

   The patch management enablement pack contains the following files:

   ```
   patches/jboss-fuse-full-6.2.0.redhat-133-baseline.zip
   system/io/fabric8/patch/patch-commands/1.2.0.redhat-621084/patch-commands-1.2.0.redhat-621084.jar
   system/io/fabric8/patch/patch-management/1.2.0.redhat-621084/patch-management-1.2.0.redhat-621084.jar
   system/io/fabric8/patch/patch-features/1.2.0.redhat-621084/patch-features-1.2.0.redhat-621084-features.xml
   system/io/fabric8/patch/patch-core/1.2.0.redhat-621084/patch-core-1.2.0.redhat-621084.jar
   ```

   NOTE

   It does not matter whether the container is running or not when you extract these files.

3. Start the container, if it is not already running.

4. Enable the new patch management commands in the old 6.2.0 container, by entering the following console commands:

   ```
   features:uninstall patch
   features:removeurl mvn:io.fabric8.patch/patch-features/1.2.0.redhat-133/xml/features
   features:addurl mvn:io.fabric8.patch/patch-features/1.2.0.redhat-621084/xml/features
   features:install patch
   ```

   The effect of this is to replace the old patch commands by the new patch commands in the existing 6.2.0 container, thereby bootstrapping the new patch mechanism (which is needed to install the 6.2.1 patch).

5. Before proceeding to the next phase, verify that the new patch commands have been successfully installed. Enter the following command:

   ```
   JBossFuse:karaf@root> list -s -t 0 | grep -i patch
   ```

```
[ 265] [Active     ] [            ] [       ] [    2]
io.fabric8.patch.patch-management (1.2.0.redhat-621084)
[ 266] [Active     ] [            ] [       ] [   80]
io.fabric8.patch.patch-core (1.2.0.redhat-621084)
[ 267] [Active     ] [            ] [       ] [   80]
io.fabric8.patch.patch-commands (1.2.0.redhat-621084)
```

Check that the preceding bundles and *only these bundles* are output by this command.

6. Add the 6.2.1 patch to the container's environment using the **patch:add** command (remembering to customize the path to the patch file), as follows:

```
JBossFuse:karaf@root> patch:add file:///path/to/jboss-fuse-full-
6.2.1.redhat-084.zip
[name]                           [installed] [description]
jboss-fuse-full-6.2.1.redhat-084 false       jboss-fuse-full-
6.2.1.redhat-084
```

NOTE

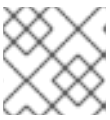
In this case, the patch file is the full distribution of JBoss Fuse 6.2.1. Under the new patching mechanism, the full distribution file has a dual purpose: you can extract the archive directly to create a fresh 6.2.1 distribution; or you can add the file as a patch to migrate an existing 6.2.0 installation to 6.2.1.

7. Simulate installing the patch using the **patch:simulate** command.

   This will generate a log of the changes that will be made to the container when the patch is installed, 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.

8. Apply the patch to the container using the **patch:install** command:

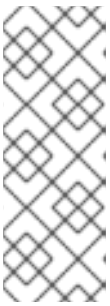
```
patch:install jboss-fuse-full-6.2.1.redhat-084
```



NOTE

Make sure that the container has fully started before you run **patch:install**.

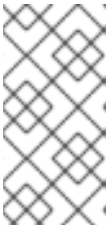
9. After installing the patch, the container shuts down automatically. At this point, the **data/cache** directory is empty, but it will be repopulated when the container starts up again.



NOTE

(Windows O/S only) At this point, it is likely that the **bin\fuse.bat** script (which calls the **bin\karaf.bat** script) will output an error as it exits. This error can be safely ignored. This happens because, on Windows, the **bin\karaf.bat** script continues to execute even after it has been overwritten by the **patch:install** command (whereas on Linux or UNIX, the corresponding script process exits immediately).

10. Open the **bin/setenv** file (**bin\setenv.bat** on Windows O/S) in a text editor and check the value of the **JAVA_MAX_MEM** environment variable, which must be set to *at least* **1024M**. Increase the setting, if necessary.



NOTE

The JBoss Fuse 6.2.1 rollup patch tries to update the **JAVA_MAX_MEM** setting to the correct value. But if you have already customized the **setenv** file yourself, the patching mechanism preserves your custom settings. This could result in the **JAVA_MAX_MEM** environment variable having a value that is too low.

11. The JBoss Fuse 6.2.1 rollup patch over-writes the contents of the **etc/org.ops4j.pax.logging.cfg** configuration file (in order fix a security issue). If you previously made any customizations to this file, they will be over-written. If this is the case, edit the file to re-apply your changes.

12. Restart the container manually, as follows:

```
./bin/fuse
```

As the container starts up again, the patching mechanism repopulates the container with all of the requisite bundles and features for the patched standalone container. When the Welcome banner appears, it displays the new patch version:

```
        _ ____                  ______
       | |  _ \                |  ____|
       | | |_) | ___  ___ ___  | |__ _   _ ___  ___
   _   | |  _ < / _ \/ __/ __| |  __| | | / __|/ _ \
  | |__| | |_) | (_) \__ \__ \ | |  | |_| \__ \  __/
   \____/|____/ \___/|___/___/ |_|   \__,_|___/\___|

    JBoss Fuse (6.2.1.redhat-084)
    http://www.redhat.com/products/jbossenterprisemiddleware/fuse/

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.

Open a browser to http://localhost:8181 to access the management
console

Create a new Fabric via 'fabric:create'
or join an existing Fabric via 'fabric:join [someUrls]'

Hit '<ctrl-d>' or 'osgi:shutdown' to shutdown JBoss Fuse.
```

TIP

If you were invoking the scripts from within the ***InstallDir*/bin** directory, you might find that the **bin/** directory appears to be empty after the container shuts down. This is because the contents of this directory were re-written by the patch. To see the scripts again, simply re-enter the **bin/** directory, for example: **cd ..;cd bin**.

4.2. PATCHING A FABRIC FROM 6.2.0 TO 6.2.1

Overview

The instructions in this section are for upgrading an existing JBoss Fuse 6.2.0 Fabric installation to version 6.2.1.

WARNING

Rolling back the patch level from version 6.2.1 to 6.2.0 is *not supported* in JBoss Fuse Fabric. This is a special case, because the version 6.2.0 Fabric agent does not support the new patching mechanism.

New patching mechanism

Upgrading from JBoss Fuse 6.2.0 to 6.2.1 requires the new patching mechanism, which is implemented for the first time in 6.2.1. This presents a bootstrapping problem: the existing 6.2.0 installation must be enhanced to support the new patching mechanism before you can install patch version 6.2.1. The upgrade procedure therefore consists of the following distinct phases:

1. Install the patch management enablement pack for 6.2.0, which replaces the existing patch mechanism with the new patching mechanism. In all other respects, the container remains at version 6.2.0.

2. Install the 6.2.1 patch in the container using the new patch mechanism and create a new profile version to store the 6.2.1 patched profiles.

3. Upgrade each of the containers in the fabric to the patched version.

Upgrading different container types

A typical fabric consists of a variety of different container types. When migrating the fabric from 6.2.0 to 6.2.1, the different container types have to be handled in slightly different ways, as follows:

Root container

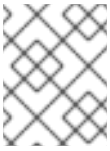
The root container is the container where you initially install the patch. The root container plays a key role in the patch process—for example, by acting as a source of patch files for the other containers in the fabric. For this reason, it is recommended that you upgrade this container last of all.

SSH container

There are some special steps required to prepare SSH containers for patching. See the section called “Preparing for a Fabric SSH container upgrade”.

Child container

Because child containers share some files and configuration with their parent container, they can easily get into an inconsistent state during the patching process. The simplest way to deal with child containers is to shut them down and focus on upgrading the parent container initially. After the parent container has been successfully upgraded, you can turn your attention to the child containers.



NOTE

Child containers cannot be kept at a lower patch version than the root container. They must be upgraded to *at least* the same patch version as the parent container.

Establishing a baseline for an SSH container upgrade

The new patching mechanism keeps track of all the changes that are made as successive patches are installed (in order to be able to roll back the patches, if needed). Hence, the first step performed by the patching mechanism is to scan the existing container installation to discover its initial state (establishing a *baseline* for subsequent changes introduced by patches). In particular, the patch mechanism scans certain subdirectories of the **`system/`** directory, to discover the initial set of bundles and Maven artifacts available in the installation.

In the case of the JBoss Fuse 6.2.0 distribution (which has not been optimised to work with the new patching mechanism), a problem arises because the core Fabric distribution, **`fabric8-karaf-1.2.0.redhat-133.zip`**, is initially *not* included with the 6.2.0 distribution and is not present in **`system/`**. When Fabric requires a copy of this file (for example, for creating a remote SSH container), it has two alternative ways of obtaining it:

- By downloading the missing core Fabric artifact, **`fabric8-karaf-1.2.0.redhat-133.zip`**, from a remote Maven repository.
- By assembling the **`fabric8-karaf-1.2.0.redhat-133.zip`** file on the fly from the contents of the root container.

The differences between these two Fabric archives are minor (for example, different branding in the welcome banner) and both are supported for use in a fabric. If it has already been created or downloaded, the Fabric distribution Zip file will be stored in the following location under **`system/`**:

```
system/io/fabric8/fabric8-karaf/1.2.0.redhat-133/fabric8-karaf-
1.2.0.redhat-133.zip
```

For establishing an initial baseline, what counts are the bundles installed in the **`system/`** directory of the root installation (the container you are using to install patches across the fabric). It can happen, however, that an SSH container uses a different version of **`fabric8-karaf-1.2.0.redhat-133.zip`** from the one that is installed in the root container. *If the Fabric distribution installed in root and the Fabric distribution installed in the SSH container are different, it is impossible to establish a proper baseline for the SSH container and patching of the SSH container will fail.*

The patching mechanism has been specially modified to enable a workaround for this problem. Specifically for the **`fabric8-karaf-1.2.0.redhat-133.zip`** Maven artifact (and only for this artifact), it is possible to install two different artifact versions under the **`system/`** directory, with the following names:

```
system/io/fabric8/fabric8-karaf/1.2.0.redhat-133/fabric8-karaf-
1.2.0.redhat-133.zip
system/io/fabric8/fabric8-karaf/1.2.0.redhat-133/fabric8-karaf-
1.2.0.redhat-133-custom.zip
```

Where one of the file names ends in **`.zip`** and the other ends in **`-custom.zip`**. It does not matter which file is which. When both of the alternative Fabric distribution files are stored in this way, it becomes possible for the patching mechanism to establish a baseline for either of the Fabric distributions and

patching of SSH containers will now work.

Initial system

The starting point for this patching procedures is assumed to be an installation of Red Hat JBoss Fuse 6.2.0 Full Install (**jboss-fuse-full-6.2.0.redhat-133.zip**), which is already configured as part of a Fabric (see).

This can be a container instance that you have customized in various ways, by adding application bundles and features, or even by editing configuration files under the **etc/** directory.

NOTE

The new patching process is usually non-destructive, preserving any customizations made before the patch was applied. If a merge conflict cannot be resolved automatically, however, warning messages will be generated in the log file.

Download the required packages

To patch JBoss Fuse from 6.2.0 to 6.2.1 you require the following packages:

patch-management-for-fuse-620-6.2.1.redhat-084.zip

Available as the download file, Red Hat JBoss Fuse 6.2.1 on Karaf Update Installer.

jboss-fuse-full-6.2.1.redhat-084.zip

Available as the download file, Red Hat JBoss Fuse 6.2.1 on Karaf Installer.

Preparing for a Fabric SSH container upgrade

(SSH containers only) Before performing any of the steps to apply the 6.2.1 patch to Fabric, prepare for SSH container upgrades by performing the following steps (for a detailed explanation of why this is necessary, see the section called “Establishing a baseline for an SSH container upgrade”):

1. In the installation of your root container, look for the following file under the **system/** directory:

   ```
   system/io/fabric8/fabric8-karaf/1.2.0.redhat-133/fabric8-karaf-
   1.2.0.redhat-133.zip
   ```

   Use a file system command to get the exact size of this file in bytes (which provides a simple way of identifying this file).

   NOTE

   It is possible that there is no **fabric8-karaf-1.2.0.redhat-133.zip** file located under this directory, which would suggest that you have not used the root container to create any SSH containers.

2. Look for the **fabric8-karaf-1.2.0.redhat-133.zip** that is installed with your SSH container (or SSH containers). Use a file system command to get the exact size of this file in bytes. Repeat this for every SSH container instance in your fabric.

3. Compare the file sizes obtained from the SSH containers with the file size obtained from the root container. If all of the file sizes are the same, this indicates that all of the containers in the Fabric are using exactly the same Fabric distribution—proceed straight to the section called “Applying the 6.2.1 patch to a Fabric container”.

   If any of the file sizes obtained from the SSH containers differ from the file size obtained from the root container, this indicates that at least one of the SSH containers is using a Fabric distribution that is different from the root container's Fabric distribution—proceed to the next step.

4. Take one of the **`fabric8-karaf-1.2.0.redhat-133.zip`** files that is *different* from the **`fabric8-karaf-1.2.0.redhat-133.zip`** file already stored under the root container's **`system/`** directory (where the difference is indicated by having a different file size), rename it to **`fabric8-karaf-1.2.0.redhat-133-custom.zip`**, and copy it to the following location under the root container's **`system/`** directory:

   ```
   system/io/fabric8/fabric8-karaf/1.2.0.redhat-133/fabric8-karaf-
   1.2.0.redhat-133-custom.zip
   ```

   When you are finished, there should be two files located under this directory, as follows:

   ```
   system/io/fabric8/fabric8-karaf/1.2.0.redhat-133/fabric8-karaf-
   1.2.0.redhat-133.zip
   system/io/fabric8/fabric8-karaf/1.2.0.redhat-133/fabric8-karaf-
   1.2.0.redhat-133-custom.zip
   ```

Applying the 6.2.1 patch to a Fabric container

To upgrade a JBoss Fuse 6.2.0 Fabric container to version 6.2.1, proceed as follows:

1. Make a full backup of your JBoss Fuse 6.2.0 installation before attempting to apply the patch.

2. Install the patch management enablement pack for 6.2.0, **`patch-management-for-fuse-620-6.2.1.redhat-084.zip`**, on top of your 6.2.0 installation. Use an archive utility to extract the contents on top of the existing 6.2.0 installation.

   The patch management enablement pack contains the following files:

   ```
   patches/jboss-fuse-full-6.2.0.redhat-133-baseline.zip
   system/io/fabric8/patch/patch-commands/1.2.0.redhat-621084/patch-
   commands-1.2.0.redhat-621084.jar
   system/io/fabric8/patch/patch-management/1.2.0.redhat-621084/patch-
   management-1.2.0.redhat-621084.jar
   system/io/fabric8/patch/patch-features/1.2.0.redhat-621084/patch-
   features-1.2.0.redhat-621084-features.xml
   system/io/fabric8/patch/patch-core/1.2.0.redhat-621084/patch-core-
   1.2.0.redhat-621084.jar
   ```

   NOTE

   It does not matter whether the container is running or not when you extract these files.

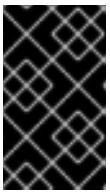
3. Start the container, if it is not already running.

4. Shut down all of the child containers in the fabric using the **container-stop** command:

```
fabric:container-stop ChildContainerList
```

5. Create a new version for the updated patch mechanism, using the **fabric:version-create** command:

```
JBossFuse:karaf@root> fabric:version-create 1.0.1
Created version: 1.0.1 as copy of: 1.0
```



IMPORTANT

The version name must be a pure *numeric* string, such as **1.0.1**, **1.1**, **2.1**, or **2.2**. You cannot incorporate alphabetic characters in the version name (such as **1.0.patch**).

6. Add the new patch feature repository to version **1.0.1** of the **default** profile, as follows:

```
fabric:profile-edit --repository mvn:io.fabric8.patch/patch-
features/1.2.0.redhat-621084/xml/features default 1.0.1
```

7. Add the **patch** and **patch-core** features to version **1.0.1** of the **default** profile, as follows:

```
fabric:profile-edit --feature patch --feature patch-core default
1.0.1
```

8. Upgrade the root container to version **1.0.1**, as follows:

```
fabric:container-upgrade 1.0.1 root
```

The effect of this upgrade is to replace the old patch commands by the new patch commands in the current 6.2.0 Fabric container, thereby bootstrapping the new patch mechanism (which is needed to install the 6.2.1 patch).

9. Wait until the current container is successfully re-provisioned to version **1.0.1** before proceeding to the next phase of the patch installation. You can monitor the provision status of the current container by entering the following console command:

```
watch container-list
```

When the **[provision status]** changes to **success**, you can proceed with the next step.

10. Add the 6.2.1 patch to the container's environment using the **patch:add** command (remembering to customize the path to the patch file), as follows:

```
JBossFuse:karaf@root> patch:add file:///path/to/jboss-fuse-full-
6.2.1.redhat-084.zip
[name]                          [installed] [description]
jboss-fuse-full-6.2.1.redhat-084 false       jboss-fuse-full-
6.2.1.redhat-084
```

NOTE

In this case, the patch file is the full distribution of JBoss Fuse 6.2.1. Under the new patching mechanism, the full distribution file has a dual purpose: you can extract the archive directly, to create a fresh 6.2.1 distribution; or you can add the file as a patch, in order to migrate an existing 6.2.0 installation to 6.2.1.

11. Create a new version, using the **`fabric:version-create`** command:

```
JBossFuse:karaf@root> fabric:version-create 1.1
Created version: 1.1 as copy of: 1.0.1
```

12. Apply the patch to the new version, **`1.1`**, using the **`patch:fabric-install`** command. Note that in order to run this command you *must* provide the credentials, ***`Username`*** and ***`Password`***, of a user with **`Administrator`** privileges. For example:

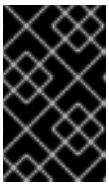
```
patch:fabric-install --username Username --password Password --
upload --version 1.1 jboss-fuse-full-6.2.1.redhat-084
```

13. Synchronize the patch information across the fabric, to ensure that the profile changes in version **`1.1`** are propagated to all containers in the fabric (particularly remote SSH containers). Enter the following console command:

```
patch:fabric-synchronize
```

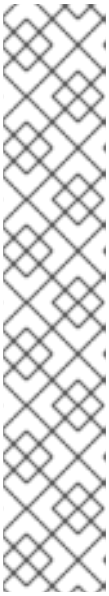
14. Upgrade each existing container in the fabric using the **`fabric:container-upgrade`** command (but leaving the root container, where you installed the patch, until last) along with its respective child containers. For example, to upgrade a container named **`remote`** and its child, **`childOfRemote`**, enter the following command:

```
fabric:container-upgrade 1.1 remote childOfRemote
```



IMPORTANT

It is recommended that you initially upgrade only one or two containers to the patched profile version, to ensure that the patch does not introduce any new issues.



NOTE

If the upgraded remote container gives the following error:

```
Provision error:
io.fabric8.common.util.MultiException: Error restarting
bundles        at
...
Caused by:
org.eclipse.jgit.api.errors.JGitInternalException:
Invalid ref name: baseline-ssh-fabric8-1.2.0.redhat-133
...
```

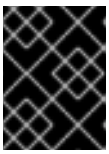
This implies that you omitted to follow the steps required to prepare for upgrading an SSH container—see the section called “Preparing for a Fabric SSH container upgrade”.

15. Keep checking the provision status of the container you are upgrading until the status appears as **`requires full restart`**. Enter the **`fabric:container-list`** command to check the status, as follows:

```
fabric:container-list
```

NOTE

After the target container has been upgraded to the patch version, the target container requires a full restart. The restart *cannot* be performed automatically by the patching mechanism, because the auto-restart capability of the patching mechanism will not become available until after the restart.



IMPORTANT

Do not attempt to restart the container you are upgrading until the status appears as **`requires full restart`**.

16. Use one of the standard mechanisms to stop and restart the container manually. In some cases, it will be possible to do this using Fabric commands from the console of the root container.

    For example, you could stop the **`remote`** container as follows:

```
fabric:container-stop remote
```

    And restart the **`remote`** container as follows:

```
fabric:container-start remote
```

17. Wait until the provision status of the container you are upgrading appears as **`success`** and then start up its child containers (if any). For example, to restart the **`childOfRemote`** container:

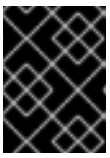
```
fabric:container-start childOfRemote
```

18. Upgrade the root container last (that is, the container that you originally installed the patch on) and its children (if any). For example, to upgrade the root container, **root**, and its child, **childOfRoot**, enter the following command:

```
fabric:container-upgrade 1.1 root childOfRoot
```

19. Keep checking the provision status of the root container until the status appears as **requires full restart**. Enter the **fabric:container-list** command to check the status, as follows:

```
fabric:container-list
```



IMPORTANT

Do not attempt to restart the root container until the status appears as **requires full restart**.

20. The root container must also be restarted manually. Shut it down using the **shutdown** console command, as follows:

```
JBossFuse:karaf@root> shutdown
Confirm: shutdown instance root (yes/no): yes
```

Restart the container manually, as follows:

```
./bin/fuse
```

TIP

If you were invoking the scripts from within the ***InstallDir*/bin** directory, you might find that the **bin/** directory appears to be empty after the container shuts down. This is because the contents of this directory were re-written by the patch. To see the scripts again, simply re-enter the **bin/** directory, for example: **cd ..;cd bin**.

21. Wait until the provision status of the root container appears as **success** and then start up its child containers (if any). For example, to restart the **childOfRoot** container:

```
fabric:container-start childOfRoot
```

22. Now set the default profile version to **1.1** (the version that has the 6.2.1 patch applied):

```
fabric:version-set-default 1.1
```

This ensures that when you create new containers from now on, those containers will use the version **1.1** profiles by default (otherwise you would have to specify version **1.1** explicitly in the container create command).

23. The JBoss Fuse 6.2.1 rollup patch over-writes the properties from the **org.ops4j.pax.logging** persistent ID (PID) in the **karaf** profile (in order to fix a security issue). If you previously made any customizations to this logging PID, they will be over-written. If

this is the case, edit the **karaf** profile to re-apply your changes—for example, by invoking the built-in profile text editor in the Karaf console, as follows:

```
profile-edit --pid org.ops4j.pax.logging karaf 1.1
```

Rolling back the 6.2.1 patch in a Fabric container

It is *not* possible to roll back the 6.2.1 patch in a Fabric container. Specifically, if you applied the 6.2.1 patch as described here, then rolling back from profile version **1.1** to version **1.0.1** (using the **fabric:container-rollback** command) is guaranteed to fail; and rolling back from profile version **1.1** to version **1.0** is also guaranteed to fail.

WARNING

Rolling back the patch level from version 6.2.1 to 6.2.0 is *not supported* in JBoss Fuse Fabric. This is a special case, because the version 6.2.0 Fabric agent does not support the new patching mechanism.

CHAPTER 5. MIGRATE MAVEN PROJECTS

OVERVIEW

JBoss Fuse has a JBoss Fuse BOM (Bill of Materials), which defines the versions of all the JBoss Fuse Maven artifacts. You can use the BOM to simplify migration of your Maven POM files. Instead of updating the **`version`** elements on each Maven dependency, all you need to do is to import the latest JBoss Fuse BOM, which defines default versions for all of the dependencies provided by JBoss Fuse.

JBOSS FUSE BOM

The JBoss Fuse BOM is a parent POM that defines the versions for all of the Maven artifacts provided by JBoss Fuse. The JBoss Fuse BOM exploits Maven's *dependency management* mechanism to specify default versions for the Maven artifacts, so that it is no longer necessary to specify the artifact versions explicitly in your POM.

CURRENT VERSION OF THE JBOSS FUSE BOM

The easiest way to discover the current version of the JBoss Fuse BOM is to examine one of the sample **`pom.xml`** files from the **`quickstarts`** examples. For example, in the ***`InstallDir`*`/quickstarts/eip/pom.xml`** file, you can find a line that defines the JBoss Fuse BOM version, as follows:

```
<project ...>
    ...
    <properties>
        ...
        <!-- the version of the JBoss Fuse BOM, defining all the
dependency versions -->
        <jboss.fuse.bom.version>6.2.1.redhat-084</jboss.fuse.bom.version>
    </properties>
    ...
</project>
```

HOW TO MIGRATE MAVEN DEPENDENCIES USING THE JBOSS FUSE BOM

To migrate Maven dependencies using the JBoss Fuse BOM, open the Maven **`pom.xml`** file for your project and edit it as follows:

1. Define the JBoss Fuse BOM version as a property in your **`pom.xml`** file, using the current BOM version. For example:

```
<project ...>
    ...
    <properties>
        ...
        <jboss.fuse.bom.version>6.2.1.redhat-
084</jboss.fuse.bom.version>
    </properties>
    ...
</project>
```

2. Reference the JBoss Fuse BOM parent POM in a **dependencyManagement** element, so that it defines default versions for the artifacts provided by JBoss Fuse. Add the following **dependencyManagement** element to your **pom.xml** file:

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

3. Now delete all of the **version** elements in your JBoss Fuse dependencies. All of the versions defined in the JBoss Fuse BOM will be applied automatically to your dependencies (through the Maven dependency management mechanism). For example, if you already had some Apache Camel dependencies, as follows:

```
 <dependencies>
   <dependency>
     <groupId>org.apache.camel</groupId>
     <artifactId>camel-core</artifactId>
<version>2.15.1.redhat-621084</version>
   </dependency>
   <dependency>
     <groupId>org.apache.camel</groupId>
     <artifactId>camel-blueprint</artifactId>
<version>2.15.1.redhat-621084</version>
   </dependency>
   <dependency>
     <groupId>org.apache.camel</groupId>
     <artifactId>camel-jetty</artifactId>
<version>2.15.1.redhat-621084</version>
   </dependency>
   ...
 </dependencies>
```

You would delete the version elements, so that the dependencies are specified as follows:

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

```
    </dependency>
    <dependency>
      <groupId>org.apache.camel</groupId>
      <artifactId>camel-jetty</artifactId>
    </dependency>
    ...
  </dependencies>
```

4. In future, when you migrate to a later version of JBoss Fuse, all that you need to do to upgrade your **pom.xml** file is to edit the **jboss.fuse.bom.version** property, so that it references the new version of the JBoss Fuse BOM.

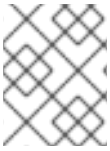
CHAPTER 6. MIGRATE DATA STORE

OVERVIEW

JBoss Fuse on Apache Karaf uses a KahaDB data store. There is an automatic migration facility that enables the KahaDB data store to be migrated to the new JBoss Fuse version.

The Aries transaction module must be installed and enabled before it can be used. See Fuse Transaction Guide for more details. Ignore the Aries transaction files instructions below if you do not have Aries installed.

MIGRATE THE KAHADB DATA STORE



NOTE

When migrating or patching JBoss Fuse, always back up the KahaDB files and Aries transaction files.

1. Backup the KhahaDB files and Aries transaction files from the old container. The files can be found at:
   - KahaDB files - ***InstallDir*/data/amq/kahadb/*.***
   - Aries transaction files - ***InstallDir*/data/txlog/*.***
2. Manually copy all of the KahaDB files from the old container to the same location in the new container.
3. Manually copy all Aries transaction log files from the same location in the sold container to the new container.

Auto-migration will take place when the new container is started.