Red Hat JBoss A-MQ 6.2

Migration Guide

Migrating to Red Hat JBoss A-MQ 6.2

Last Updated: 2018-06-15

Red Hat JBoss A-MQ 6.2 Migration Guide

Migrating to Red Hat JBoss A-MQ 6.2

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

Legal Notice

Abstract

This guide lays out the issues a user will encounter when upgrading to the latest version of Red Hat JBoss A-MQ.

Table of Contents

CHAPTER 1. MIGRATION OVERVIEW

Abstract

This chapter highlights some of the key points that might affect your applications, when migrating from JBoss A-MQ 6.1 to JBoss A-MQ 6.2.

1.1. UPGRADED COMPONENTS

Version upgrades

Many of the major components in JBoss Fuse and JBoss A-MQ 6.2 have been upgraded. The following versions are used in JBoss A-MQ:

Table 1.1. Component Versions

Component	Version for 6.1	Version for 6.2
Apache ActiveMQ	5.9.x	5.11.0
Apache Karaf	2.3.x	2.4.0
Fabric8 (was Fuse Fabric)	1.0.0	1.2.0
Spring framework	3.2.x	3.2.x

Apache ActiveMQ changes

The changes resulting from the upgrade to version 5.11.0 are detailed in Chapter 4, *Apache ActiveMQ Issues*.

1.2. NEW ROLE-BASED ACCESS CONTROL

Overview

JBoss A-MQ 6.2 has a new role-based access control (RBAC) feature, which is enabled by default in the container. The new RBAC system offers differentiated access to the container, depending on which roles have been assigned to a user. The RBAC imposes access controls on the Karaf console (so that only administrators can access the full range of commands and command options) and imposes access controls on the JMX protocol (so that access control is applied to the Fuse Management Console).

Standardized roles

Table 1.2, “Standard Roles for Access Control” lists and describes the standard roles that are used throughout the JMX ACLs and the command console ACLs.

Table 1.2. Standard Roles for Access Control

Roles	Description
`Monitor`, **`Operator`**, **`Maintainer`**	Grants read-only access to the container.
`Deployer`, **`Auditor`**	Grants read-write access at the appropriate level for ordinary users, who want to deploy and run applications. But blocks access to sensitive container configuration settings.
`Administrator`, **`SuperUser`**	Grants unrestricted access to the container.

Migrating user data for RBAC

When migrating to JBoss A-MQ 6.2, you must modify your user data, so that users are assigned one of the standard roles from Table 1.2, “Standard Roles for Access Control”.

Reference

For more details about role-based access control, see section "Role-Based Access Control" in "Security Guide".

1.3. SECURITY CHANGES

Overriding the default JAAS realm in Fabric

In JBoss A-MQ 6.2, the rank of the default **`ZookeeperLoginModule`** JAAS module (which is installed by default in a Fabric container) has changed to **`99`**, and the name of the default realm is**`karaf`**. In previous releases, the rank of **`ZookeeperLoginModule`** realm was just **`1`**.

Hence, if you want to override the default **`karaf`** in the context of Fabric, you must define a new realm named **`karaf`**, with a **`rank`** attribute that is greater than or equal to **`100`**.

Enabling LDAP authentication in a Fabric

In particular, this change affects the configuration needed to enable LDAP authentication in a Fabric. In this case, the **`rank`** attribute of the **`jaas:config`** element in the JAAS realm configuration file must be increased to at least **`100`** (recommended is **`200`**). For details, see section "Procedure for a Fabric" in "Security Guide".

1.4. MIGRATING SPRING-DM TO BLUEPRINT

Spring-DM is now deprecated

Spring Dynamic Modules (Spring-DM) is now *deprecated* and will be removed from a future release of JBoss A-MQ. You can continue to use Spring XML and the Spring framework, however, as long as you do not invoke the Spring-DM component.

Prefer Blueprint over Spring-DM

The Blueprint container is now the preferred framework for instantiating, registering, and referencing OSGi services, because this container has now been adopted as an OSGi standard. This ensures greater portability for your OSGi service definitions in the future.

Spring Dynamic Modules (Spring-DM) provided much of the original impetus for the definition of the Blueprint standard, but should now be regarded as obsolescent. Using the Blueprint container does *not* prevent you from using the Spring framework: the latest version of Spring is compatible with Blueprint.

How to tell whether your code uses Spring-DM

In Spring XML files, the Spring-DM component is associated with the following XML namespace:

```
http://www.springframework.org/schema/osgi
```

To identify the parts of your application that use Spring-DM, search for the preceding namespace string in your code.

How to migrate Spring-DM to Blueprint

If you have a Spring XML file that uses the Spring-DM component, migrate this file to Blueprint XML, as follows:

1. In the standard Maven project layout, Blueprint XML files are stored under the following directory:

   ```
   src/main/resources/OSGI-INF/blueprint
   ```

   If it does not already exist, create this directory under your Maven project.

2. Move the relevant Spring XML file from the Spring directory, **`src/main/resources/META-INF/spring`**, to the Blueprint directory, **`src/main/resources/OSGI-INF/blueprint`**.

3. Edit the Spring XML file in order to convert it to a Blueprint XML file. For example, a typical Spring XML file using Spring-DM has the following outline:

   ```
   <beans xmlns="http://www.springframework.org/schema/beans"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:osgi="http://www.springframework.org/schema/osgi">
       ...
       <osgi:reference id="osgiPlatformTransactionManager"

   interface="org.springframework.transaction.PlatformTransactionManage
   r"/>

       <osgi:reference id="osgiJtaTransactionManager"

   interface="javax.transaction.TransactionManager"/>
       ...
   </beans>
   ```

   You can convert this Spring XML file to a Blueprint XML file by replacing the **`beans`** root element by a root **`blueprint`** root element, and replacing Spring-DM **`osgi:reference`** elements by Blueprint **`reference`** elements. For example:

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    ...
    <reference id="osgiPlatformTransactionManager"

interface="org.springframework.transaction.PlatformTransactionManage
r"/>

    <reference id="osgiJtaTransactionManager"
               interface="javax.transaction.TransactionManager"/>
    ...
</blueprint>
```

1.5. MIGRATING MAVEN PROJECTS

Overview

JBoss A-MQ 6.2 now has a JBoss A-MQ BOM (Bill of Materials), which defines the versions of all the JBoss A-MQ 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 A-MQ BOM, which defines default versions for all of the dependencies provided by JBoss A-MQ.

JBos A-MQ BOM

The JBoss A-MQ BOM is a parent POM that defines the versions for all of the Maven artifacts provided by JBoss A-MQ. The JBoss A-MQ 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 JBos A-MQ BOM

The easiest way to discover the current version of the JBoss A-MQ 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 A-MQ BOM version, as follows:

```
<project ...>
    ...
    <properties>
        ...
        <!-- the version of the JBoss A-MQ 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 JBos A-MQ BOM

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

1. Define the JBoss A-MQ 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 A-MQ BOM parent POM in a **dependencyManagement** element, so that it defines default versions for the artifacts provided by JBoss A-MQ. 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 A-MQ dependencies. All of the versions defined in the JBoss A-MQ 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 A-MQ, 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 A-MQ BOM.

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.

SPRING DYNAMIC MODULES (SPRING-DM) IS DEPRECATED

Spring-DM (which integrates Spring XML with the OSGi service layer) is deprecated in 6.2 and you should use the Blueprint framework instead. Using Blueprint does not prevent you from using the Spring framework: the latest version of Spring is compatible with Blueprint.

SERVICEMIX MAVEN ARCHETYPES NOT SUPPORTED

The ServiceMix Maven archetypes (with a **`groupId`** of **`org.apache.servicemix.tooling`**) are no longer supported and are not available in 6.2. You can use the fabric8 Maven archetypes instead (which provide similar functionality). The fabric8 archetypes have a **`groupId`** of **`io.fabric8.archetypes`** and the following fabric8 archetypes are available:

```
karaf-camel-amq-archetype
karaf-camel-cbr-archetype
karaf-camel-cxf-code-first-archetype
karaf-camel-cxf-contract-first-archetype
karaf-camel-dozer-wiki-archetype
karaf-camel-drools-archetype
karaf-camel-eips-archetype
karaf-camel-errorhandler-archetype
karaf-camel-log-archetype
karaf-camel-log-wiki-archetype
karaf-camel-webservice-archetype
karaf-rest-archetype
karaf-secure-rest-archetype
karaf-secure-soap-archetype
karaf-soap-archetype
```

FUSE APPLICATION BUNDLES

Fuse Application Bundles (FABs) are no longer supported and are not available in 6.2. Instead of using FABs, it is recommended that you repackage your code as an OSGi bundle, for deployment into the JBoss A-MQ container.

CHAPTER 3. CONSOLE CHANGES

FABRIC:MQ-CREATE COMMAND

The following argument names have changed in JBoss A-MQ 6.2:

- **`--ports`** to **`--port`**
- **`--networks`** to **`--network`**

FABRIC:PROFILE-CREATE COMMAND

The following argument names have changed in JBoss A-MQ 6.2:

- **`--parents`** to **`--parent`**

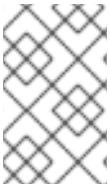
FABRIC:PROFILE-EDIT COMMAND

The following argument names have changed in JBoss A-MQ 6.2:

- **`--repositories`** to **`--repository`**
- **`--features`** to **`--feature`**
- **`--libs`** to **`--lib`**
- **`--bundles`** to **`--bundle`**

FABRIC:EXPORT AND FABRIC:IMPORT COMMANDS

The **`fabric:export`** and **`fabric:import`** commands have been removed in JBoss A-MQ 6.2, and are now replaced by the corresponding **`zk:export`** and **`zk:import`** commands. To gain access to these Zookeeper commands, you must install the **`fabric-zookeeper-commands`** feature.



NOTE

The **`zk:export`** and **`zk:import`** commands interact purely with the Zookeeper registry. For example, you cannot use these commands to export or import Fabric profile data, which is now stored in the container's Git repository.

CHAPTER 4. APACHE ACTIVEMQ ISSUES

Abstract

JBoss A-MQ 6.2 uses Apache ActiveMQ 5.11.0. Since the last release, Apache ActiveMQ has been upgraded from version 5.9.0 to version 5.11.0. This introduces a few migration issues.

4.1. KEY MIGRATION ISSUES

JMS ObjectMessage serialization

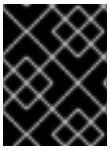
If you are migrating an application that uses JMS ObjectMessage serialization (that is, sending or receiving messages containing a serialized Java object using the **`javax.jms.ObjectMessage`** type), in JBoss A-MQ 6.2.1 you must now set the **`org.apache.activemq.SERIALIZABLE_PACKAGES`** JVM property. The **`org.apache.activemq.SERIALIZABLE_PACKAGES`** property must be set to the list of Java packages that are allowed to be serialized.

For example, on the broker side, you can set this property in the ***`InstallDir`*`/etc/system.properties`** file, as follows:

```
org.apache.activemq.SERIALIZABLE_PACKAGES="java.lang,java.util,org.apache.
activemq,org.fusesource.hawtbuf,com.thoughtworks.xstream.mapper,com.mycomp
any.myapp"
```

If you do not set this JVM property, you are liable to see an error message like the following after migrating your application to JBoss A-MQ 6.2.1:

```
Caused by: javax.jms.JMSException: Failed to build body from content.
Serializable class not available to broker. Reason:
java.lang.ClassNotFoundException: Forbidden class
com.sundar.verify.MyBean! This class is not allowed to be serialized. Add
package with 'org.apache.activemq.SERIALIZABLE_PACKAGES' system property.
```



IMPORTANT

JMS ObjectMessage serialization poses significant security risks. If you use this feature, make sure that you understand the risks—see Security in Object Serialization.

4.2. MIGRATING CLIENTS

Migrating Apache ActiveMQ clients

In general, it is recommended that you update your Apache ActiveMQ clients at the same time that you update the brokers, in order to guarantee compatibility between clients and brokers.

It is possible, in some cases, that older client versions might be interoperable with later broker versions. The Openwire protocol supports version negotiation, such that an old client can negotiate the lowest common version with its peer and use that version. But JBoss A-MQ does not have a comprehensive test suite for testing compatibility between all of the different versions of Apache ActiveMQ. Hence, to be sure of compatibility, it is recommended that you upgrade your clients along with your brokers to use the same version.

4.3. NEW FEATURES

ActiveMQ 5.10.0

In ActiveMQ 5.10.0, the following new features have been introduced:

- Hardened MQTT support
- Hardened AMQP support
- Hardened LevelDB store
- Improved RAR/JCA adapter
- Improved Runtime configuration plugin

ActiveMQ 5.11.0

In ActiveMQ 5.11.0, the following new features have been introduced:

- Destination import/export for lock down mode. Use the **`destinationsPlugin`** on the broker to import/export broker destinations to s a specified location. For example:

```
<plugins>
    <destinationsPlugin location="/workspace/destinations"/>
</plugins>
```

- Dynamic Camel root loading. Allows routes to modified on the fly without restarting the broker. To take advantage of this feature, you must define a **`camelroutesBrokerPlugin`** plug-in in the broker configuration, as follows:

```
<plugins>
    <camelroutesBrokerPlugin routesFile="routes.xml" />
</plugins>
```

  Where the **`routes.xml`** file must be in the same location as the broker configuration file.

- MQTT: QOS2 mapped to virtual topics. Can be enabled using the transport option, **`?transport.subscriptionStrategy="mqtt-virtual-topic-subscriptions"`**.
- Start scripts simplification
- Recover scheduler database option

4.4. DEPENDENCY UPGRADES

Spring framework

JBoss A-MQ and JBoss Fuse use Spring framework version 3.2.

Apache Karaf

JBoss A-MQ and JBoss Fuse use Apache Karaf version 2.4.0.

4.5. API CHANGES

JMS streams

JMS streams are now *deprecated*. If you need to send very large messages, it is recommended that you use an out-of-bounds transport, such as FTP, instead. In particular, the **`org.apache.activemq.ActiveMQInputStream`**and **`ActiveMQOutputStream`** classes are deprecated, as are the **`ActiveMQConnection.createInputStream`** and **`ActiveMQConnection.createOutputStream`** methods.

Camel ActiveMQ component

Removed the **`org.apache.activemq.camel.converter.IdentityMessageReuseConverter`** class from the Camel ActiveMQ component (**`activemq-camel`**).

CHAPTER 5. MIGRATING JBOSS A-MQ FROM 6.2.0 TO 6.2.1 ON KARAF

Abstract

If you have already deployed an instance of JBoss A-MQ 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.

5.1. 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 A-MQ 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 A-MQ 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 A-MQ 6.2.0 (**`jboss-a-mq-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 A-MQ from 6.2.0 to 6.2.1 you require the following packages:

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

Available as the download file, Red Hat JBoss A-MQ 6.2.1 on Karaf Update Installer.

`jboss-a-mq-6.2.1.redhat-084.zip`

Available as the download file, Red Hat JBoss A-MQ 6.2.1.

Applying the 6.2.1 patch to a standalone container

To upgrade a JBoss A-MQ 6.2.0 standalone container to version 6.2.1, proceed as follows:

1. Make a full backup of your JBoss A-MQ 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-amq-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-a-mq-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-core/1.2.0.redhat-621084/patch-core-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-management/1.2.0.redhat-621084/patch-management-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:

```
JBossA-MQ:karaf@root> patch:add file:///path/to/jboss-a-mq-
6.2.1.redhat-084.zip
[name]                      [installed] [description]
jboss-a-mq-6.2.1.redhat-084 false       jboss-a-mq-6.2.1.redhat-084
```

NOTE

In this case, the patch file is the full distribution of JBoss A-MQ 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-a-mq-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 A-MQ 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 A-MQ 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/amq
```

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 A-MQ (6.2.1.redhat-084)
    http://www.redhat.com/products/jbossenterprisemiddleware/amq/

  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

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

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**.

5.2. PATCHING A FABRIC FROM 6.2.0 TO 6.2.1

Overview

The instructions in this section are for upgrading an existing JBoss A-MQ 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 A-MQ Fabric. This is a special case, because the version 6.2.0 Fabric agent does not support the new patching mechanism.

Prerequisite for patching Fabric in JBoss A-MQ

The following artifact—which is required in order to patch a Fabric with the new patching mechanism—is missing from the **`jboss-a-mq-6.2.1.redhat-084.zip`** distribution:

```
system/io/fabric8/fabric8-karaf/1.2.0.redhat-621084/fabric8-karaf-
1.2.0.redhat-621084.zip
```

Consequently, even after adding the JBoss A-MQ 6.2.1 rollup patch to your existing 6.2.0 container, this file will be missing from your 6.2.0 container installation.

This file is *required* in order to patch a JBoss A-MQ Fabric system. The missing file will be provided in the first patch for JBoss A-MQ. In the meantime, if you have access to the Red Hat JBoss Fuse 6.2.1 on Karaf Update Installer distribution (**`jboss-fuse-full-6.2.1.redhat-084.zip`**), you can copy the missing file from that distribution into the location given above. Otherwise, please contact support.

New patching mechanism

Upgrading from JBoss A-MQ 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 A-MQ 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:

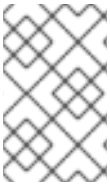
```
system/io/fabric8/fabric8-karaf/1.2.0.redhat-133/fabric8-karaf-
1.2.0.redhat-133.zip
system/io/fabric8/fabric8-karaf/1.2.0.redhat-133/fabric8-karaf-
1.2.0.redhat-133-custom.zip
```

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 A-MQ 6.2.0 (**jboss-a-mq-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 A-MQ from 6.2.0 to 6.2.1 you require the following packages:

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

Available as the download file, Red Hat JBoss A-MQ 6.2.1 on Karaf Update Installer.

jboss-a-mq-6.2.1.redhat-084.zip

Available as the download file, Red Hat JBoss A-MQ 6.2.1.

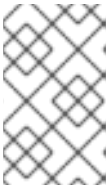
Preparing for a Fabric SSH container upgrade

(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 A-MQ 6.2.0 Fabric container to version 6.2.1, proceed as follows:

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

2. Install the patch management enablement pack for 6.2.0, **`patch-management-for-amq-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-a-mq-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-core/1.2.0.redhat-621084/patch-core-
   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-management/1.2.0.redhat-621084/patch-
management-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).

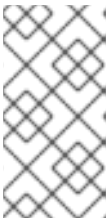
9. Wait until the current container is successfully re-provisioned to version **`1.0.1`** before proceeding to the next phase of the patch installation. You can monitor the provision status of the current container by entering the following console command:

```
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:

```
JBossA-MQ:karaf@root> patch:add file:///path/to/jboss-a-mq-
6.2.1.redhat-084.zip
[name]                       [installed] [description]
jboss-a-mq-6.2.1.redhat-084 false       jboss-a-mq-6.2.1.redhat-084
```



NOTE

In this case, the patch file is the full distribution of JBoss A-MQ 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:

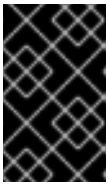
```
patch:fabric-install --username Username --password Password --
upload --version 1.1 jboss-a-mq-6.2.1.redhat-084
```

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:

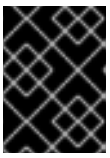
```
fabric:container-start childOfRemote
```

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/amq
```

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 A-MQ 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 A-MQ Fabric. This is a special case, because the version 6.2.0 Fabric agent does not support the new patching mechanism.

CHAPTER 6. MIGRATE DATA STORE

OVERVIEW

JBoss A-MQ 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 A-MQ 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 A-MQ, 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.