Red Hat JBoss A-MQ 6.1

Migration Guide

Migrating to Red Hat JBoss A-MQ 6.1

Last Updated: 2017-10-13

Red Hat JBoss A-MQ 6.1 Migration Guide

Migrating to Red Hat JBoss A-MQ 6.1

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 6.0 to JBoss A-MQ 6.1. For a detailed description of the changes made to each of the components, see the relevant chapters for Apache ActiveMQ, Apache Camel, and Apache CXF.

1.1. UPGRADED COMPONENTS

Version upgrades

Many of the major components in JBoss Fuse and JBoss A-MQ 6.1 have been upgraded. The following versions are used in JBoss A-MQ:

Table 1.1. Component Versions

Component	Version for 6.0	Version for 6.1
Apache ActiveMQ	5.8.0	5.9.x
Apache Karaf	2.3.0	2.3.x
Fabric8 (was Fuse Fabric)	7.2.0	1.0.0
Spring framework	3.1.3	3.2.x

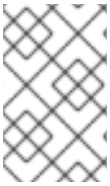
Fuse Management Console

In JBoss A-MQ 6.1, the implementation of the Fuse Management Console has changed and it is now based on the open source Hawt.io project. The new Fuse Management Console has a radically different look-and-feel from the old version, with greatly expanded functionality and many new features.

When you start up the JBoss A-MQ container (for example, by running the command, **`bin/amq`**), the new Fuse Management Console is installed by default. You can access it by navigating to the following URL in your browser:

```
http://localhost:8181
```

For more details, see "Management Console User Guide".



NOTE

The ActiveMQ Web console has also been removed from the JBoss A-MQ 6.1 product. Equivalent functionality for Apache ActiveMQ is now provided by the Fuse Management Console.

Fuse IDE

The Fuse Integrated Development Environment (IDE) is no longer provided as a standalone Eclipse

executable. Fuse IDE has been renamed to *Fuse Plugins* for JBoss Developer Studio. To install Fuse Plugins, first download and install JBoss Developer Studio and then install the relevant Eclipse plug-ins. For detailed instructions, see "Installation Guide".

ActiveMQ resource adapter on JBoss EWS, JBoss EAP, and JBoss Fuse Service Works

The ActiveMQ resource adapter has been certified to function correctly on the JBoss Enterprise Web Services container, the JBoss Enterprise Application Platform container, and on JBoss Fuse Service Works, enabling you integrate JMS applications running on those containers with JBoss A-MQ.

Apache Camel deployed on JBoss EWS and JBoss EAP

The Apache Camel core library has been certified to function correctly on the JBoss Enterprise Web Services container and on the JBoss Enterprise Application Platform container.

Apache ActiveMQ changes

The changes resulting from the upgrade to version 5.9.0 are detailed in Chapter 2, *Apache ActiveMQ Issues*.

The most important changes are:

- The ActiveMQ Web console is no longer supported.
- If you are using a JDBC store with a broker that uses XA transactions, you will need to make some changes to the database schema, when migrating the database store.
- Dynamic configuration updates are now supported on a standalone broker.
- A generic JMS XA connection pool library is now available, which makes it possible to deploy third-party JMS providers.

Apache Karaf Web console

The Karaf Web console is no longer supported in JBoss A-MQ 6.1. You can use the new Hawtio-based Fuse Management Console instead, which has the same functionality (and more) as the Karaf Web console.

1.2. 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>
```

1.3. FABRIC MIGRATION

Overview

If you use Fuse Fabric containers to deploy your applications, you should read this section to understand the issues that affect migration to JBoss A-MQ. If you use standalone containers, however, you can ignore this section.

Fabric package has changed

In JBoss Fuse 6.1 and JBoss A-MQ 6.1, the Fabric package name has changed from **`org.fusesource.fabric`** to **`io.fabric8`**, which has the following migration impact:

- Configuration PIDs for Fabric have been renamed from **`org.fusesource.fabric.`*`name`*** to **`io.fabric8.`*`name`***. For example, this affects the names of the following configuration files under the **`etc/`** directory:

  ```
  io.fabric8.datastore.cfg
  io.fabric8.fab.osgi.url.cfg
  io.fabric8.maven.cfg
  ```

- If you use any Fabric APIs directly in your own application code, you must replace references to the **`org.fusesource.fabric`** package name by **`io.fabric8`**.

- If your Maven **`pom.xml`** files have any direct dependencies on Fabric artifacts, you need to change the Maven GroupId from **`org.fusesource.fabric`** to **`io.fabric8`** (as well as updating the dependency version). For example, the following **`fabric-zookeeper`** dependency:

  ```
  <dependency>
      <groupId>org.fusesource.fabric</groupId>
      <artifactId>fabric-zookeeper</artifactId>
      <version>7.2.0</version>
  </dependency>
  ```

  Should be replaced by the following dependency:

  ```
  <dependency>
      <groupId>io.fabric8</groupId>
      <artifactId>fabric-zookeeper</artifactId>
      <version>1.0.0.redhat-379</version>
  </dependency>
  ```

1.4. MIGRATING MAVEN PROJECTS

Overview

JBoss A-MQ 6.1 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.

JBoss 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 JBoss 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>6.1.0.redhat-379</jboss.fuse.bom.version>
    </properties>
    ...
</project>
```

How to migrate Maven dependencies using the JBoss 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>6.1.0.redhat-
   379</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>2.12.0.redhat-610379</version>
   </dependency>
   <dependency>
     <groupId>org.apache.camel</groupId>
     <artifactId>camel-blueprint</artifactId>
<version>2.12.0.redhat-610379</version>
   </dependency>
   <dependency>
     <groupId>org.apache.camel</groupId>
     <artifactId>camel-jetty</artifactId>
<version>2.12.0.redhat-610379</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. APACHE ACTIVEMQ ISSUES

Abstract

JBoss A-MQ 6.1.0.redhat-379 uses Apache ActiveMQ 5.9.0. Since the last release, Apache ActiveMQ has been upgraded from version 5.8.0 to version 5.9.0. This introduces a few migration issues.

2.1. KEY MIGRATION ISSUES

ActiveMQ Web console no longer supported

The Apache ActiveMQ Web console is no longer supported in JBoss A-MQ 6.1 and JBoss Fuse 6.1. It is recommended that you use the Fuse Management Console instead (installed by default in the container). You can access the new Hawtio-based Fuse Management Console by navigating to the **`http://localhost:8181`** URL in your browser.

Migrating a JDBC store with XA transactions

If you are using the JDBC persistence adapter with XA transactions enabled, you should note that the database schema for the JDBC store has changed, going from version 5.8.0 to version 5.9.0. If you need to migrate the contents of an existing JDBC store to ActiveMQ 5.9.0, you must perform the following steps:

1. Note that this change to the database schema only affects JDBC stores that used with XA transactions. If your existing broker does not use XA transactions, there is no need to perform these migration steps.
2. If the old broker (version 5.8.0) is currently running, make sure no transactions are in flight before attempting to stop the broker.
3. Stop the broker.
4. Alter the database tables by executing database commands similar to the following (you will probably need to adapt these commands to suit the syntax of your particular database):

```
ALTER TABLE ACTIVEMQ_MSGS ALTER COLUMN XID VARCHAR(250)
CREATE INDEX ACTIVEMQ_MSGS_XIDX ON ACTIVEMQ_MSGS (XID)
ALTER TABLE ACTIVEMQ_ACKS ALTER COLUMN XID VARCHAR(250)
CREATE INDEX ACTIVEMQ_ACKS_XIDX ON ACTIVEMQ_ACKS (XID)
```

5. Configure the new broker (version 5.9.0) to use the JDBC store.
6. Start the new broker.

fabric:mq-create command has different default profile name

In JBoss A-MQ version 6.0, the **`fabric:mq-create`** command would create a new profile with the same name as the broker, by default. In JBoss A-MQ 6.1, however, this behaviour has changed, so that the default profile name is **`mq-broker-`*`Group`*`.`*`BrokerName`***, where ***`Group`*** is the broker group and ***`BrokerName`*** is the name of the new broker.

If you want to get the same behaviour as in version 6.0, you can use the **--profile** option to specify the new profile name explicitly. For example, to create a new JBoss A-MQ broker, where both the broker and the profile are named **brokerx**, enter a command like the following:

```
fabric:mq-create --profile brokerx --create-container broker --replicas 1
--networks us-west brokerx
```

fabric:mq-create command requires keytool utility

The **fabric:mq-create** command requires that the Java **keytool** command-line utility is provided on your PATH. If necessary, you can work around this requirement by specifying the **--no-ssl** option (which disables support for the SSL/TLS protocol).

2.2. 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.

2.3. NEW FEATURES

Dynamic configuration updates

JBoss A-MQ now has a runtime configuration plug-in, which enables you to update the configuration of a standalone broker (that is, a broker that is not part of a fabric) dynamically at run time.

For details, see the section "Modifying a Running Standalone Broker's XML Configuration" in the chapter "Editing a Broker's Configuration" in the *Managing and Monitoring a Broker* guide from the JBoss A-MQ library.

Generic JMS XA connection pool

The **activemq-jms-pool** component is a new library that provides generic JMS connection pool classes that support the XA transaction protocol. Using this library, it is possible for third-party JMS providers to participate in XA transactions managed by any JTA transaction manager.

For details of the generic XA-aware connection pool library, see the chapter "XA Transactions in Red Hat JBoss Fuse" in the *Transaction Guide* from the JBoss Fuse library.

AMQP protocol

Red Hat JBoss Fuse 6.1 now supports the AMQP 1.0 protocol (OASIS Advanced Message Queuing Protocol). You can access the AMQP protocol using a URI of the form, **amqp://*Hostname*:*Port***. SSL security is also supported.

Red Hat JBoss Fuse 6.1 includes the Apache Qpid client API, which provides a JMS client API that is compatible with AMQP. You can use this client API to implement AMQP clients in Java.

NOTE

AMQP 1.0 is *not* a supported protocol for the JBoss A-MQ JCA connector (Apache ActiveMQ resource adapter, which is the plug-in for integrating JBoss A-MQ with the JBoss Enterprise Application Platform container). OpenWire is the only wire protocol supported by the JCA connector.

C, C++ and .Net client libraries for Openwire

Red Hat JBoss Fuse 6.1 includes the C, C++ (CMS) client library and the .Net (NMS) client library for Openwire (the native Apache ActiveMQ wire protocol).

LevelDB store on Linux platforms

The new LevelDB store is now supported *on Linux platforms only*.

2.4. FEATURE PREVIEWS

Technical previews of features in development

The following features are included in the JBoss Fuse 6.1 release in order to let you preview technology that is currently under development. *These features are not yet supported and are not suitable for deployment in a production environment.*

LevelDB store (on selected platforms)

Red Hat JBoss Fuse includes a technical preview of the new LevelDB store, which uses Google's LevelDB library to maintain the indexes in the log files. The LevelDB store supports XA transactions in ActiveMQ 5.9. The LevelDB store is available only on Windows, Linux, and Mac OS platforms.

NOTE

The LevelDB store is provided with native libraries on Linux only. On all of the other supported operating systems, the LevelDB technical preview is provided as a Java implementation.

IMPORTANT

On Windows and Mac OS platforms, this is a technical preview only and is not suitable for deployment in a production environment. The LevelDB store is currently supported *only* on the Red Hat Enterprise Linux (RHEL) platform (for the RHEL versions and configurations specified in *Supported Configurations*).

2.5. SECURITY FIXES

SSL transport cipher suites

Fixed AMQ-4582, affecting the SSL transport cipher suites. Previously, if you specified an invalid cipher suite to the **`transport.enabledCipherSuites`** parameter on an SSL transport connector, the broker would start with all ciphers enabled.

JAAS authorization now compatible with Karaf JAAS authentication

The implementation of the JAAS authorization plug-in has been modified so that it is compatible with the Apache Karaf JAAS authentication module. This makes it possible to integrate the JAAS authorization plug-in with the Karaf JAAS authentication module when the broker is deployed in an OSGi container. For more details, see the *JBoss A-MQ Security Guide*.

Allow Bouncy Castle security provider to be used

Fixed AMQ-4520, which is caused by a bug in the default SSL provider that comes with Java 7 (affecting the Diffie-Hellman cipher suite). You can now work around this issue by adding the Bouncy Castle security producer to the Java 7 **`lib`** directory.

Removed command agent

Removed the command agent, which is no longer needed and might potentially have exposed a security hole through the JMS protocol.

2.6. PERSISTENCE AND FAILOVER

Recommended message stores

If you are using one of the deprecated message stores, it is recommended that you migrate to one of the following message stores instead:

- KahaDB message store.
- JDBC message store.

Multiple stores for different destinations

See AMQ-4310.

2.7. TRANSPORT PROTOCOL CHANGES

XMPP transport

The XMPP transport is now *deprecated* and will be removed in a future release.

Openwire over UDP

The Openwire over UDP protocol combination is now *deprecated* and will be removed in a future release.

2.8. DEPENDENCY UPGRADES

Spring framework

JBoss A-MQ and JBoss Fuse use Spring framework version 3.2.x. Spring version 3.0.x and version 3.1.x *not* supported in this release.

Apache Karaf

JBoss A-MQ and JBoss Fuse use Apache Karaf version 2.3.0.

Jasypt

In ActiveMQ 5.9.0, Jasypt is upgraded to version 1.9.1.

Jolokia

In ActiveMQ 5.9.0, Jolokia is upgraded to version 1.1.4.

Apache Xerces

In ActiveMQ 5.9.0, Apache Xerces is upgraded to version 2.11.0.

json-simple

In ActiveMQ 5.9.0, **`json-simple`** is upgraded to version 1.1.1.

Apache Derby

In ActiveMQ 5.9.0, Apache Derby is upgraded to version 10.10.1.1.

Apache commons-io

In ActiveMQ 5.9.0, Apache commons-io is upgraded to version 2.4.

Apache Qpid

In ActiveMQ 5.9.0, Apache Qpid is upgraded to version 0.26.

2.9. API CHANGES

isSlave method reinstated

In Apache ActiveMQ 5.9.0, reinstated the **`isSlave`** method on the **`org.apache.activemq.broker.BrokerService`** class.

PooledConnectionFactory

Renamed the **`setTimeBetweenExpirationCheckMillis`** method to **`getTimeBetweenExpirationCheckMillis`** in the **`org.apache.activemq.jms.pool.PooledConnectionFactory`** class.

New activemq-spring module

In ActiveMQ 5.9.0, the code base has been refactored to localize dependencies on the Spring framework, which are now packaged in the new **`activemq-spring`** module.