Red Hat JBoss A-MQ

Integrating with JBoss Enterprise Application Platform

Version 6.1
April 2014

Integrating with JBoss Enterprise Application Platform

Version 6.1

Updated: 28 Apr 2014

Copyright © 2011-2014 Red Hat, Inc. and/or its affiliates.

Trademark Disclaimer

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

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

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

Apache, ServiceMix, Camel, CXF, and ActiveMQ are trademarks of Apache Software Foundation. Any other names contained herein may be trademarks of their respective owners.

Third Party Acknowledgements

One or more products in the Red Hat JBoss A-MQ release includes third party components covered by licenses that require that the following documentation notices be provided:

- JLine (http://jline.sourceforge.net) jline:jline:jar:1.0

  License: BSD (LICENSE.txt) - Copyright (c) 2002-2006, Marc Prud'hommeaux `<mwp1@cornell.edu>`

  All rights reserved.

  Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

  - Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
  - Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
  - Neither the name of JLine nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

- Stax2 API (http://woodstox.codehaus.org/StAX2) org.codehaus.woodstox:stax2-api:jar:3.1.1

  License: The BSD License (http://www.opensource.org/licenses/bsd-license.php)

  Copyright (c) <YEAR>, <OWNER> All rights reserved.

  Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

  - Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
  - Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

- jibx-run - JiBX runtime (http://www.jibx.org/main-reactor/jibx-run) org.jibx:jibx-run:bundle:1.2.3

  License: BSD (http://jibx.sourceforge.net/jibx-license.html) Copyright (c) 2003-2010, Dennis M. Sosnoski.

  All rights reserved.

  Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

  - Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
  - Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
  - Neither the name of JiBX nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON

ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

- JavaAssist (http://www.jboss.org/javassist) org.jboss.javassist:com.springsource.javassist:jar:3.9.0.GA:compile

  License: MPL (http://www.mozilla.org/MPL/MPL-1.1.html)

- HAPI-OSGI-Base Module (http://hl7api.sourceforge.net/hapi-osgi-base/) ca.uhn.hapi:hapi-osgi-base:bundle:1.2

  License: Mozilla Public License 1.1 (http://www.mozilla.org/MPL/MPL-1.1.txt)

Table of Contents

1. Deploying the Apache ActiveMQ Resource Adapter 9
Install the ActiveMQ Resource Adapter 10
Integrating with an ActiveMQ Failover Cluster 19
Install JBoss AS Quickstarts 22
Build and Deploy the helloworld-mdb Example 27

List of Examples

1.1. ActiveMQ Resource Adapter Configuration in standalone.xml 12
1.2. Message Driven Bean Configuration in standalone.xml 14
1.3. ActiveMQ Resource Adapter Configuration for Failover 20

Chapter 1. Deploying the Apache ActiveMQ Resource Adapter

This chapter explains how to install the Apache ActiveMQ resource adapter into JBoss Enterprise Application Platform and how to integrate ActiveMQ messaging into your applications, taking the `helloworld-mdb` demonstration as an example.

Install the ActiveMQ Resource Adapter 10
Integrating with an ActiveMQ Failover Cluster 19
Install JBoss AS Quickstarts 22
Build and Deploy the helloworld-mdb Example 27

Install the ActiveMQ Resource Adapter

Overview

This section describes how to find, install, and configure the ActiveMQ resource adapter into a standalone instance of the JBoss Enterprise Application Platform.

A *resource adapter* is a kind of plug-in for a J2EE container. The J2EE standard defines the resource adapter framework, which makes it possible to expand the core J2EE container, adding new features and functionality. By installing the ActiveMQ resource adapter, you make it possible for message driven beans and servlets to communicate through an external Apache ActiveMQ broker instance. The ActiveMQ broker can thus be used as the underlying messaging system in the container.

Resource adapter location

You can find the ActiveMQ resource adapter archive file, `activemq-rar-5.9.0.redhat-610379.rar`, at either of the following locations:

- In the following Zip archive file:

  ```
  InstallDir/extras/apache-activemq-5.9.0.redhat-610379-bin.zip
  ```

  After expanding the archive, the resource adapter file can be found in the following sub-directory:

  ```
  apache-activemq-5.9.0.redhat-610379/lib/optional
  ```

- Directly from the Red Hat JBoss Fuse Maven repository, at the following URL:

  ```
  http://repo.fusesource.com/nexus/content/groups/public/org/apache/activemq/activemq-rar/
  ```

  Download the `.rar` archive file from the appropriately versioned sub-directory, 5.9.0.redhat-610379.

Configuration files

The following configuration files are needed for the the ActiveMQ resource adapter (when installed in a standalone instance of the JBoss Enterprise Application Platform):

InstallDir`/standalone/configuration/standalone.xml`
: The `standalone.xml` file is the default (bare bones) configuration for the JBoss Enterprise Application Platform container. You must edit this file to complete the installation of the ActiveMQ resource adapter.

Note

It is assumed that this file does *not* already configure the HornetQ messaging system (which would conflict with the ActiveMQ messaging system).

Note

JBoss Enterprise Application Platform can be figured either as a standalone container, using `standalone/configuration/standalone.xml`, or as a managed domain, using `domain/configuration/domain.xml`. Throughout this section, we describe explicitly how to configure the standalone container, but it is understood that a similar approach could be used to configure a managed domain.

Steps to install the resource adapter

Perform the following steps to install the Apache ActiveMQ resource adapter into JBoss Enterprise Application Platform (assuming that you will be running the container in standalone mode):

1. Extract the Apache ActiveMQ community distribution. You can find an archive of the Apache ActiveMQ distribution in the following location:

   ```
   InstallDir/extras/apache-activemq-5.9.0.redhat-610379-bin.zip
   ```

   Using a suitable archive utility, extract the preceding archive file to any convenient location on your filesystem. The root of the extracted directory tree is called `apache-activemq-5.9.0.redhat-610379` by default.

2. The ActiveMQ resource adapter archive file can now be found under the `/lib/optional` sub-directory of the archive extracted in the previous step. Make a copy of the ActiveMQ resource adapter archive file, omitting the version number in the filename. For example, on a UNIX or Linux platform, you can rename the `activemq-rar-5.9.0.redhat-610379.rar` archive file as follows:

   ```
   cd apache-activemq-5.9.0.redhat-610379/lib/optional
   cp activemq-rar-5.9.0.redhat-610379.rar activemq-rar.rar
   ```

Note

Renaming the resource adapter archive in this way is not strictly necessary. But because the resource adapter file name appears in the resource adapter configuration, using a versionless filename makes it easier to upgrade the resource adapter at a later date.

3. Install the ActiveMQ resource adapter by copying the resource adapter archive, `activemq-rar.rar`, to the relevant JBoss Enterprise Application Platform deployment directory. For example, on a UNIX or Linux platform, you could copy the resource adapter archive to a standalone JBoss Enterprise Application Platform as follows:

```
cp activemq-rar.rar EAPInstallDir/standalone/deployments/
```

4. Add the requisite resource adapter configuration to the `urn:jboss:domain:resource-adapters:1.1` subsystem in the JBoss Enterprise Application Platform configuration, as follows.

   Open the *`EAPInstallDir`*`/standalone/configuration/standalone.xml` file using a text editor and paste the `resource-adapter` element from Example 1.1 on page 12 into the `urn:jboss:domain:resource-adapters:1.1` subsystem, as a child of the `resource-adapters` element.

   Example 1.1. ActiveMQ Resource Adapter Configuration in standalone.xml

```
<server xmlns="urn:jboss:domain:1.4">
    ...
    <profile>
        ...
        <subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">
            <resource-adapters>
                <resource-adapter id="activemq-rar.rar">
                    <archive>
                        activemq-rar.rar
                    </archive>
                    <transaction-support>XATransaction</transaction-support>
                    <config-property name="UserName">
                        defaultUser
                    </config-property>
                    <config-property name="Password">
                        defaultPassword
                    </config-property>
                    <config-property name="ServerUrl">
                        tcp://localhost:61616?jms.rmIdFromConnectionId=true
                    </config-property>
                    <connection-definitions>
```

```
                        <connection-definition
                              class-name="org.apache.activemq.ra.ActiveMQManagedConnec
tionFactory"
                              jndi-name="java:/ConnectionFactory"
                              enabled="true"
                              pool-name="ConnectionFactory">
                            <xa-pool>
                                <min-pool-size>1</min-pool-size>
                                <max-pool-size>20</max-pool-size>
                                <prefill>false</prefill>
                                <is-same-rm-override>false</is-same-rm-override>
                            </xa-pool>
                        </connection-definition>
                    </connection-definitions>
                    <admin-objects>
                        <admin-object
                              class-name="org.apache.activemq.command.ActiveMQQueue"
                              jndi-name="java:/queue/HELLOWORLDMDBQueue"
                              use-java-context="true"
                              pool-name="HELLOWORLDMDBQueue">
                            <config-property name="PhysicalName">
                                HELLOWORLDMDBQueue
                            </config-property>
                        </admin-object>
                        <admin-object
                              class-name="org.apache.activemq.command.ActiveMQTopic"
                              jndi-name="java:/topic/HELLOWORLDMDBTopic"
                              use-java-context="true"
                              pool-name="HELLOWORLDMDBTopic">
                            <config-property name="PhysicalName">
                                HELLOWORLDMDBTopic
                            </config-property>
                        </admin-object>
                    </admin-objects>
                </resource-adapter>
                ...
            </resource-adapters>
        </subsystem>
        ...
    </profile>
    ...
</server>
```

If your resource adapter archive filename differs from `activemq-rar.rar`, you must change the content of the `archive` element in the preceding configuration to match the name of your archive file.

The values of the `UserName` and `Password` configuration properties must be chosen to match the credentials of a valid user in the external broker.

You might need to change the value of the `ServerUrl` configuration property to match the actual hostname and port exposed by the external broker.

Important

In order to ensure that JMS transactions are integrated correctly, it is essential to include the `jms.rmIdFromConnectionId=true` option setting on the `ServerUrl` configuration property and to include the `<is-same-rm-override>false</is-same-rm-override>` setting in the `xa-pool` element, as shown above.

Note

The JMS administrative objects defined in the `admin-objects` element do not need to be defined yet. They serve as examples to show how you can define administrative objects for the ActiveMQ resource adapter (they are used later in the message-driven bean demonstration).

5. Add the requisite message driven bean configuration to the `urn:jboss:domain:ejb3:1.4` subsystem in the JBoss Enterprise Application Platform configuration.

   Open the *`EAPInstallDir`*`/standalone/configuration/standalone.xml` file using a text editor and paste the `mdb` element from Example 1.2 on page 14 into the `urn:jboss:domain:ejb3:1.4` subsystem.

   Example 1.2. Message Driven Bean Configuration in standalone.xml

```
<server xmlns="urn:jboss:domain:1.4">
    ...
    <profile>
        ...
        <subsystem xmlns="urn:jboss:domain:ejb3:1.4">
            ...
            <mdb>
                <resource-adapter-ref resource-adapter-name="activemq-rar"/>
                <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
            </mdb>
            <pools>
                <bean-instance-pools>
                    <strict-max-pool name="slsb-strict-max-pool" max-pool-size="20" in
stance-acquisition-timeout="5" instance-acquisition-timeout-unit="MINUTES"/>
                    <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20" in
stance-acquisition-timeout="5" instance-acquisition-timeout-unit="MINUTES"/>
                </bean-instance-pools>
```

```
            </pools>
            ...
        </subsystem>
        ...
    </profile>
    ...
</server>
```

6. Before starting the broker, check the broker configuration to make sure that there are valid user credentials defined in the broker's *InstallDir*/etc/users.properties file. For example, to match the UserName and Password credentials configured in Example 1.1 on page 12, the users.properties file should contain an entry like the following:

```
defaultUser=defaultPassword,admin
```

7. Start the external A-MQ broker. For example, on a UNIX or Linux platform, you can start the JBoss A-MQ broker instance as follows:

```
cd nstallDir/bin
./fuse
```

8. Start the standalone instance of JBoss Enterprise Application Platform. For example, on a UNIX or Linux platform, you can start the standalone instance as follows:

```
cd EAPInstallDir/bin
./standalone.sh
```

Verify the installation with JConsole

You can verify the installation of the ActiveMQ resource adapter, using the Java JConsole utility, as follows:

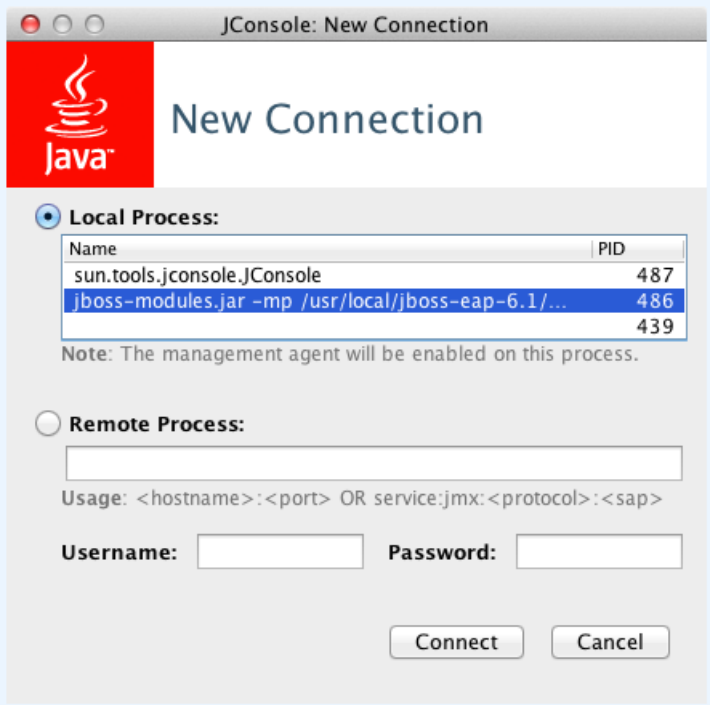
1. Make sure that the standalone instance of JBoss Enterprise Application Platform is running. If necessary, start it from the command line, as follows (UNIX or Linux platform):

```
./standalone.sh
```

2. Open a new command prompt and start up the Java JConsole utility by entering the following command:

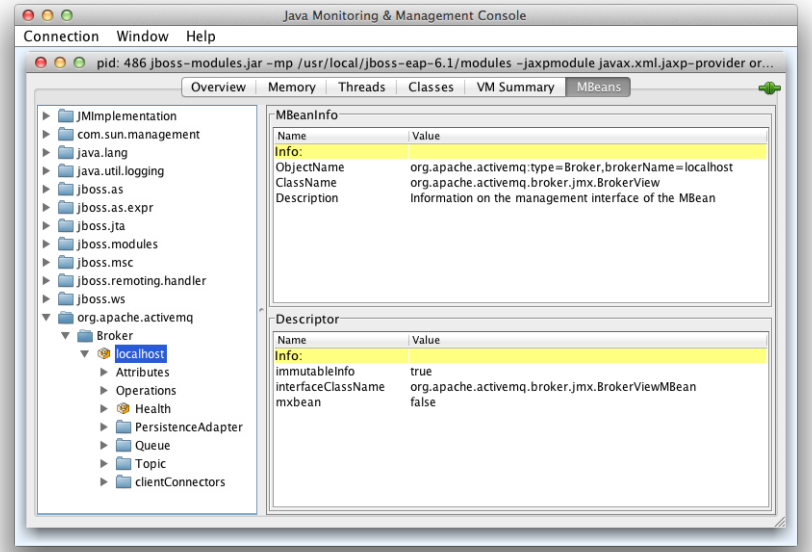
```
jconsole
```

3. The **JConsole: New Connection** dialog opens. In the **Local Process** list, select the jboss-modules.jar entry, as shown in the screeen shot, and then click **Connect**.



4. The **Java Monitoring & Management Console** window opens. In this window, select the **MBeans** tab and look for the `org.apache.activemq` folder. If the `org.apache.activemq` folder is *not* visible, this would indicate that the ActiveMQ broker is not running, and therefore there is a problem with the installation of the ActiveMQ resource adapter.

   If the `org.apache.activemq` folder is visible, you can drill down from this node to examine the MBeans in the ActiveMQ broker, as shown in the followin screen shot.



Resource adapter configuration

In the configuration shown in Example 1.1 on page 12, you use the `config-property` element to set resource adapter properties. The ActiveMQ resource adapter supports the following basic properties:

`UserName`
: *(Optional)* Specifies the client username when connecting to the ActiveMQ broker (not required in this example, because the ActiveMQ broker configuration does not enable authentication).

`Password`
: *(Optional)* Specifies the client password when connecting to the ActiveMQ broker (not required in this example, because the ActiveMQ broker configuration does not enable authentication).

`ServerUrl`
: Specifies the URL used to connect to the embedded ActiveMQ broker instance. This value must match one of the endpoints specified by a `transportConnector` element in the ActiveMQ broker configuration.

`BrokerXmlConfig`
: *(Optional)* Specifies the location of an embedded ActiveMQ broker's XML configuration file. To specify a location on the file system, use the format, `xbean:file:///AbsolutePath`, where the path, *`AbsolutePath`*, should be specified as an absolute pathname.

`UseInboundSession`
: *(Optional)* Sets a flag that specifies whether outbound connections should reuse the inbound connection's session for sending messages (useful for connections going through a firewall). Defaults to `false`.

`Clientid`
: *(Optional)* Specifies a JMS client ID, which the resource adapter uses for the connection from the JBoss Enterprise Application Platform container.

ActiveMQ broker configuration

Most of the options for customizing the ActiveMQ resource adapter are provided by the ActiveMQ broker configuration file, `broker-config.xml` (whose location is specified by the `BrokerXmlConfig` resource adapter property). This configuration file supports a huge range of features and settings which are beyond the scope of this guide. To learn more about ActiveMQ broker configuration, see the following guides from the *Red Hat JBoss A-MQ* documentation library:

- *Configuring Broker Persistence*
- *Tuning Guide*
- *Security Guide*
- *XML Configuration Reference*

Integrating with an ActiveMQ Failover Cluster

Overview

This section describes how to configure the ActiveMQ resource adapter to connect to an ActiveMQ failover cluster (for example, a high-availability master/slave cluster). For details about how to set up and configure such a cluster, see *Fault Tolerant Messaging*.

Failover URL

To connect to a cluster of ActiveMQ brokers (for example, a master/slave pair of brokers), you need to configure the `ServerUrl` configuration property with a failover URL, which lists the available endpoints in the cluster. The general form of the failover URL you should use is as follows:

```
failover:(uri1,...,uriN)?maxReconnectAttempts=0
```

Note

It is important to set the option `maxReconnectAttempts=0`, in order to ensure a clean cutover when the master fails in a master/slave high-availability cluster.

Sample scenario

Consider the scenario where a broker running on host `amqhostA` and a broker running on host `amqhostB` are configured to run as a high-availability master/slave cluster. In this scenario, the brokers expose the following TCP endpoints:

```
tcp://amqhostA:61616
tcp://amqhostB:61616
```

To connect to this cluster, the resource adapter should be configured with the following failover URL:

```
failover:(tcp://amqhostA:61616,tcp://amqhostB:61616)?jms.rmIdFromConnectionId=true&maxRecon
nectAttempts=0
```

When setting the URL in an XML file, you must remember to escape the `&` symbol as `&` giving the URL:

```
failover:(tcp://amqhostA:61616,tcp://amqhostB:61616)?jms.rmIdFromConnectionId=true&amp;maxRe
connectAttempts=0
```

Configuring the ActiveMQ resource adapter for failover

To configure the ActiveMQ resource adapter to connect to an ActiveMQ failover cluster, you must modify the following configuration settings:

- Set the `ServerUrl` configuration property to a correctly configured failover URL,
- Set the `UseInboundSession` configuration property to `true` for inbound connections (set as the direct child of the `resource-adapter` element), and
- Set the `UseInboundSession` configuration property to `false` for the connection factories (set as the child of a `connection-definition` element).

Open the *`EAPInstallDir`*`/standalone/configuration/standalone.xml` file using a text editor, search for the `urn:jboss:domain:resource-adapters:1.1` subsystem, and modify the `ServerUrl` property and the `UseInboundSession` property as shown in Example 1.3 on page 20. You will need to customize the value of the failover URL, as appropriate, to match the configuration of your broker cluster.

Example 1.3. ActiveMQ Resource Adapter Configuration for Failover

```
<server xmlns="urn:jboss:domain:1.4">
    ...
    <profile>
        ...
        <subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">
            <resource-adapters>
                <resource-adapter id="activemq-rar.rar">
                    ...
                    <config-property name="ServerUrl">
                        failover:(tcp://amqhostA:61616,tcp://amqhostB:61616)?jms.rmIdFrom
ConnectionId=true&amp;maxReconnectAttempts=0
                    </config-property>
                    ...
                    <config-property name="UseInboundSession">
                        true
                    </config-property>
                    ...
                    <connection-definitions>
                        <connection-definition ... >
                            ...
                            <config-property name="UseInboundSession">
                                false
                            </config-property>
                            ...
                        </connection-definition>
                    </connection-definitions>
                    ...
```

```
                </resource-adapter>
                ...
            </resource-adapters>
        </subsystem>
        ...
    </profile>
    ...
</server>
```

Install JBoss AS Quickstarts

Overview

The JBoss AS Quickstart examples consists of a collection of demonstrations that illustrate features of the JBoss Enterprise Application Platform. The installation consists of the following parts:

- *JBoss EAP Maven repository*—an offline Maven repository for JBoss Enterprise Application Platform, which contains the dependencies required by the quickstart examples.
- *JBoss AS Quickstart examples*—the quickstart examples themselves.

Prerequisites

To download, install, and build the JBoss AS Quickstart examples, you need the following prerequisites:

- *Subscription*—you must have a Red Hat subscription that includes support for the JBoss Enterprise Application Platform product (or ask Red Hat support for access as part of an evaluation).
- *Maven installation*—you must have Apache Maven[1] installed and the version must be 3.0.0 or later. You can get the latest copy of Maven from the Maven download page[2].
- *Internet access*—Maven is a distributed build system, which downloads packages from the Internet on the fly, whenever they are needed during a build. Consequently, you must have access to the Internet while performing a Maven build.

JBoss AS Quickstarts download location

You can download the JBoss AS Quickstart examples from the Quickstarts download page[3] on the Red Hat Customer Portal[4] site. Click the following link to download the `jboss-eap-6.1.0-quickstarts.zip` file:

- JBoss Enterprise Application Platform 6.1.0 Quickstarts[5]

[1] https://www.apache.org/
[2] https://maven.apache.org/download.cgi
[3] https://access.redhat.com/jbossnetwork/restricted/softwareDetail.html?softwareId=21733&product=appplatform&version=6.1.0&downloadType=distributions
[4] https://access.redhat.com/downloads/
[5] https://access.redhat.com/jbossnetwork/restricted/softwareDownload.html?softwareId=21733

Note

After following this link, you will be prompted to log on to the Red Hat customer access portal. If you do not have a subscription for JBoss Enterprise Application Platform, you will not be able to access this download, however.

Maven repository download location

The JBoss Enterprise Application Platform Maven repository is required in order to run the quickstart examples.

You can download the Maven repository from the Maven Repository download page[6] on the Red Hat Customer Portal[7] site. Click the following link to download the `jboss-eap-6.1.0-maven-repository.zip` file:

- JBoss Enterprise Application Platform 6.1.0 Maven Repository[8]

Steps to install JBoss AS Quickstarts

To install the JBoss AS Quickstart examples, perform the following steps:

1. Download the `jboss-eap-6.1.0-quickstarts.zip` file from the customer portal site. Use an archive utility to unzip the downloaded file at a convenient location on your filesystem, *`QuickInstallDir`*.

2. Download the `jboss-eap-6.1.0-maven-repository.zip` file from the customer portal site. Use an archive utility to unzip the downloaded file at a convenient location on your filesystem, *`MvnRepoInstallDir`*.

Note

It is essential to download and install the Maven repository on your local machine. The quickstart examples require Maven artifacts that are *not* available from any public repositories online. You will not be able to build the quickstart examples unless you download, install, and configure the Maven repository.

3. Configure Maven to use the downloaded Maven repository by editing your local repository's `settings.xml` file (usually located at `~/.m2/settings.xml` on Linux and UNIX systems, or at `C:\Documents and`

[6] https://access.redhat.com/jbossnetwork/restricted/softwareDetail.html?softwareId=21743&product=appplatform&version=6.1.0&downloadType=distributions
[7] https://access.redhat.com/downloads/
[8] https://access.redhat.com/jbossnetwork/restricted/softwareDownload.html?softwareId=21743

`Settings\Username\.m2\settings.xml` on Windows). Open the `settings.xml` file with a text editor and add the following profiles:

```
<?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 ht
tp://maven.apache.org/xsd/settings-1.0.0.xsd">

  <profiles>

    <!-- Configure the JBoss EAP Maven repository -->
    <profile>
      <id>jboss-eap-repository</id>
      <repositories>
        <repository>
          <id>jboss-eap-repository</id>
          <url>file:///path/to/jboss-eap-6.1.0.GA-maven-repository</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>jboss-eap-plugin-repository</id>
          <url>file:///path/to/jboss-eap-6.1.0.GA-maven-repository</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>

    <!-- Configure the JBoss Community Maven repository -->
    <profile>
      <id>jboss-community-repository</id>
      <repositories>
        <repository>
          <id>jboss-community-repository</id>
          <url>http://repository.jboss.org/nexus/content/groups/public/</url>
          <releases>
```

```
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>jboss-community-plugin-repository</id>
          <url>http://repository.jboss.org/nexus/content/groups/public/</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>

  </profiles>

  <activeProfiles>
    <!-- Optionally, make the repositories active by default -->
    <activeProfile>jboss-eap-repository</activeProfile>
    <activeProfile>jboss-community-repository</activeProfile>
  </activeProfiles>

</settings>
```

Note

Alternatively, there is a sample settings file provided at *`MvnRepoInstallDir`*`/example-settings.xml` in the downloaded Maven repository, which you can use as a template for defining your `settings.xml` file.

4. Replace all occurrences of `file:///path/to/jboss-eap-6.1.0.GA-maven-repository` in the `settings.xml` file with the actual location of the Maven repository on your filesystem, *`MvnRepoInstallDir`*.

Test the installation

To test the installation of the quickstart examples, try to build the `helloworld-mdb` example using Maven. Open a new command window, change directory to *`QuickInstallDir`*`/helloworld-mdb`, and enter the following command:

```
mvn clean package
```

If the project builds successfully, you should see a `BUILD SUCCESS` status and the generated `jboss-as-helloworld-mdb.war` package will be found under the *`QuickInstallDir`*`/target` directory.

If the project does not build successfully, make sure that you have access to the Internet and check that the Maven `settings.xml` file is correctly configured.

Build and Deploy the helloworld-mdb Example

Overview

In this tutorial, you will customize the `helloworld-mdb` quickstart example so that it works with the ActiveMQ resource adapter. You can then build and deploy the `helloworld-mdb` example into a standalone instance of JBoss Enterprise Application Platform (which already has an ActiveMQ resource adapter installed).

The `helloworld-mdb` example illustrates two kinds of integration with messaging: integration of message-driven beans; and integration of a servlet (which gets access to a JMS queue and a JMS topic using the standard JMS administered objects mechanism).

Prerequisites

The following prerequisites must be satisfied, before you can build and deploy the `helloworld-mdb` example:

- The ActiveMQ resource adapter is installed in the JBoss Enterprise Application Platform (as described in "Install the ActiveMQ Resource Adapter" on page 10), and the installation has been verified.
- The JBoss EAP Maven repository and the JBoss AS Quickstart examples have been installed (as described in "Install JBoss AS Quickstarts" on page 22).
- You have Internet access (for the Maven build).

Customizations

The version of the `helloworld-mdb` demonstration provided in the quickstarts distribution is integrated with the HornetQ messaging platform by default. To refactor the demonstration so that it integrates with Apache ActiveMQ, it is necessary to customize the following aspects of the `helloworld-mdb` code:

- Delete the HornetQ XML configuration file (located in `helloworld-mdb/src/webapp/WEB-INF/hornetq-jms.xml`).
- In `HelloWorldQueueMDB.java`, customize the annotations on the message driven bean to integrate with the ActiveMQ resource adapter.
- In `HelloWorldTopicMDB.java`, customize the annotations on the message driven bean to integrate with the ActiveMQ resource adapter.
- Add additional Maven dependencies.

These customizations are described in more detail in the rest of this section.

Steps to build and deploy the example

To build and deploy the quickstart `helloworld-mdb` example, perform the following steps:

1. Delete the following HornetQ XML configuration file from the `helloworld-mdb` project:

   ```
   helloworld-mdb/src/webapp/WEB-INF/hornetq-jms.xml
   ```

2. Edit the annotations on the `HelloWorldQueueMDB` message driven bean class, so that it integrates with the ActiveMQ resource adapter (instead of HornetQ). Edit the `HelloWorldQueueMDB.java` file at the following location:

   ```
   helloworld-mdb/src/main/java/org/jboss/as/quickstarts/mdb/HelloWorldQueueMDB.java
   ```

   Open the `HelloWorldQueueMDB.java` file using a text editor and make the modifications highlighted in the following extract:

   ```
   import org.jboss.ejb3.annotation.ResourceAdapter;
   ...
   @MessageDriven(name = "HelloWorldQueueMDB", activationConfig = {
     @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
   "javax.jms.Queue"),
     @ActivationConfigProperty(propertyName = "destination", propertyValue = "HELLOWORLDM
   DBQueue"),
     @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue = "Auto-ac
   knowledge") })
   @ResourceAdapter(value="activemq-rar.rar")
   public class HelloWorldQueueMDB implements MessageListener {
       ...
   ```

   Where the following changes are made to the code:

   - The `@ResourceAdapter` annotation explicitly associates the message driven bean with the ActiveMQ resource adapter. You *must* include this annotation, if you want to use the ActiveMQ resource adapter.
   - You need to add an `import` statement for the `@ResourceAdapter` annotation.
   - The value of the `destination` property is changed to `HELLOWORLDMDBQueue`, which is the *physical name* of the corresponding ActiveMQ queue that this message driven bean subscribes to. The physical name of the queue is the queue identifier used by the ActiveMQ broker.

> **Note**
> You must specify the queue's physical name here. In contrast to the case of HornetQ, the ActiveMQ messaging integration does *not* allow you to use a JNDI name for the `destination` value.

3. Edit the annotations on the `HelloWorldTopicMDB` message driven bean class, so that it integrates with the ActiveMQ resource adapter (instead of HornetQ). Edit the `HelloWorldTopicMDB.java` file at the following location:

   ```
   helloworld-mdb/src/main/java/org/jboss/as/quickstarts/mdb/HelloWorldTopicMDB.java
   ```

   Open the `HelloWorldTopicMDB.java` file using a text editor and make the modifications highlighted in the following extract:

   ```
   import org.jboss.ejb3.annotation.ResourceAdapter;
   ...
   @MessageDriven(name = "HelloWorldQTopicMDB", activationConfig = {
     @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
   "javax.jms.Topic"),
     @ActivationConfigProperty(propertyName = "destination", propertyValue = "HELLOWORLDM
   DBTopic"),
     @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue = "Auto-ac
   knowledge") })
   @ResourceAdapter(value="activemq-rar.rar")
   public class HelloWorldTopicMDB implements MessageListener {
       ...
   ```

4. Add the Maven dependency for the `jboss-ejb3-ext-api` artifact, which is needed for the `@ResourceAdapter` annotation. Open the `helloworld-mdb/pom.xml` file using a text editor and add the following `dependency` element as a child of the `dependencies` element in the POM file:

   ```
   <project ...>
       ...
       <dependencies>
           ...
           <dependency>
               <groupId>org.jboss.ejb3</groupId>
               <artifactId>jboss-ejb3-ext-api</artifactId>
               <!-- to get the right version, look in the EAP offline Maven repo -->
               <version>2.0.0-redhat-2</version>
               <scope>provided</scope>
           </dependency>
       </dependencies>
   ```

```
    ...
</project>
```

The `provided` scope value implies there is no need to embed the `jboss-ejb3-ext-api` JAR file in the generated WAR, because this library is already provided by the JBoss Enterprise Application Platform container.

Note

If you ever need to update the version of the `jboss-ejb3-ext-api` artifact, you can discover which version to use by drilling down to the `org/jboss/ejb3/jboss-ejb3-ext-api` sub-directory of the Maven repository you downloaded and installed in "Install JBoss AS Quickstarts" on page 22.

5. Build the `helloworld-mdb` example as follows. Open a new command prompt, change directory to the `helloworld-mdb` directory, and enter the following Maven command:

```
mvn clean package
```

If the build is successful, you should find the `jboss-as-helloworld-mdb.war` WAR file in the `helloworld-mdb/target` directory.

6. If you have not already done so, register the administered objects for the queues and topics used by the example, by editing the JBoss Enterprise Application Platform configuration.

In your JBoss Enterprise Application Platform installation, open the `standalone/configuration/standalone.xml` configuration file with a text editor, and add the following highlighted administered objects to the `activemq-rar.rar` resource adapter:

```
<server xmlns="urn:jboss:domain:1.4">
    ...
    <profile>
        ...
        <subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">
            <resource-adapters>
                <resource-adapter id="activemq-rar.rar">
                    ...
                    <admin-objects>
                        <admin-object
                              class-name="org.apache.activemq.command.ActiveMQQueue"
                              jndi-name="java:/queue/HELLOWORLDMDBQueue"
                              use-java-context="true"
                              pool-name="HELLOWORLDMDBQueue">
                            <config-property name="PhysicalName">
                                HELLOWORLDMDBQueue
```

```
                            </config-property>
                        </admin-object>
                        <admin-object
                              class-name="org.apache.activemq.command.ActiveMQTopic"
                              jndi-name="java:/topic/HELLOWORLDMDBTopic"
                              use-java-context="true"
                              pool-name="HELLOWORLDMDBTopic">
                            <config-property name="PhysicalName">
                                HELLOWORLDMDBTopic
                            </config-property>
                        </admin-object>
                    </admin-objects>
                </resource-adapter>
            </resource-adapters>
        </subsystem>
        ...
    </profile>
    ...
</server>
```

Where the preceding configuration adds the following entries to the JNDI registry:

`java:/queue/HELLOWORLDMDBQueue`
: References a `javax.jms.Queue` object that connects to the `HELLOWORLDMDBQueue` ActiveMQ queue (where the queue name on the ActiveMQ broker is specified by the `PhysicalName` config property).

`java:/queue/HELLOWORLDMDBTopic`
: References a `javax.jms.Topic` object that connects to the `HELLOWORLDMDBTopic` ActiveMQ topic (where the topic name on the ActiveMQ broker is specified by the `PhysicalName` config property).

In the `helloworld-mdb` example, these administered objects are accessed by the servlet class, `HelloWorldMDBServletClient` (but *not* by the message driven bean classes). For example, the `HelloWorldMDBServletClient` class injects these JNDI entries, using the `@Resource` annotation, as follows:

```
import javax.annotation.Resource;
...
import javax.jms.ConnectionFactory;
...
import javax.jms.Queue;
import javax.jms.Topic;
...
public class HelloWorldMDBServletClient extends HttpServlet {
    ...
 @Resource(mappedName = "java:/ConnectionFactory")
 private ConnectionFactory connectionFactory;

 @Resource(mappedName = "java:/queue/HELLOWORLDMDBQueue")
```

```
private Queue queue;

@Resource(mappedName = "java:/topic/HELLOWORLDMDBTopic")
private Topic topic;
   ...
```

7. Deploy the `helloworld-mdb` example to the running Web server. Manually copy the `jboss-as-helloworld-mdb.war` WAR file from the `helloworld-mdb/target` directory to the Web server's deployment directory, `standalone/deployments`.

Access the helloworld-mdb service

You can now test the `helloworld-mdb` service, as follows:

1. If you have not already started the JBoss Enterprise Application Platform standalone container, do so by entering the following commands at the command line:

```
cd EAPInstallDir/bin
./standalone.sh
```

2. You should now be able to access the `helloworld-mdb` service from your browser, by navigating to the following URL:

```
http://localhost:8080/jboss-as-helloworld-mdb/HelloWorldMDBServletClient
```

When you navigate to the preceding URL, the servlet sends five messages to the `HelloWorldQueueMDB` message-driven bean. If you look at the container console window, you should see some output like the following:

```
14:41:20,739 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldQueueMDB] (default-threads
 - 7) Received Message from queue: This is message 5
14:41:20,739 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldQueueMDB] (default-threads
 - 5) Received Message from queue: This is message 3
14:41:20,739 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldQueueMDB] (default-threads
 - 4) Received Message from queue: This is message 2
14:41:20,741 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldQueueMDB] (default-threads
 - 6) Received Message from queue: This is message 4
14:41:20,742 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldQueueMDB] (default-threads
 - 3) Received Message from queue: This is message 1
```

These console messages are written by the `HelloWorldQueueMDB` message-driven bean, thus demonstrating that the messages have successfully propagated from the servlet, through the ActiveMQ broker, to the message-driven bean.

3. To send messages to the `HelloWorldTopicMDB` message-driven bean, navigate to the following URL:

```
http://localhost:8080/jboss-as-helloworld-mdb/HelloWorldMDBServletClient?topic
```

When you navigate to the preceding URL, the servlet sends five messages to the `HelloWorldTopicMDB` message-driven bean. If you look at the container console window, you should see some output like the following:

```
14:53:02,464 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldTopicMDB] (default-threads
 - 9) Received Message from topic: This is message 2
14:53:02,466 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldTopicMDB] (default-threads
 - 10) Received Message from topic: This is message 3
14:53:02,468 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldTopicMDB] (default-threads
 - 8) Received Message from topic: This is message 1
14:53:02,471 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldTopicMDB] (default-threads
 - 11) Received Message from topic: This is message 4
14:53:02,472 INFO  [class org.jboss.as.quickstarts.mdb.HelloWorldTopicMDB] (default-threads
 - 12) Received Message from topic: This is message 5
```

These console messages are written by the `HelloWorldTopicMDB` message-driven bean, thus demonstrating that the messages have successfully propagated from the servlet, throught the ActiveMQ broker, to the message-driven bean.