JBoss Enterprise Application Platform 5

Getting Started Guide

for Use with JBoss Enterprise Application Platform 5

Edition 5.2.0

Last Updated: 2017-10-13

JBoss Enterprise Application Platform 5 Getting Started Guide

for Use with JBoss Enterprise Application Platform 5
Edition 5.2.0

Eva Kopalova

Petr Penicka

Russell Dickenson

Scott Mumford

Legal Notice

Abstract

This Getting Started Guide documents information regarding the initial use of the JBoss Enterprise Application Platform 5 and its patch releases.

Table of Contents

CHAPTER 1. OTHER CONVENTIONS

This and other JBoss Enterprise Application Platform documents make use of the following notations:

<JBOSS_HOME>

<JBOSS_HOME> refers to the directory that your instance of JBoss Enterprise Application Platform has been extracted/installed to. For example: **`/home/`*`USERNAME`*`/jboss-eap-`*`<VERSION>`*`/`**

<PROFILE>

<PROFILE> refers to the directory that contains the server profile you are making changes to. It may be a test profile or a production profile.

More information about server profiles and their locations is in Section 2.1, “Server Profiles”.

<DOMAIN_NAME>

<DOMAIN_NAME> refers to the name you have configured for your JBoss Enterprise Application Platform instance.

This name is used in some configuration files. Replace this notation, where appropriate, with the name of your domain.

CHAPTER 2. JBOSS ENTERPRISE APPLICATION PLATFORM OVERVIEW

JBoss Enterprise Application Platform 5 is an open-source JEE-based middleware solution. It is built on top of the consolidated JBoss Application Server 5 called JBoss Enterprise Application Server, which introduces the JBoss Microcontainer.

The JBoss architecture essentially consists of the microcontainer, bootstrap beans loaded into the microcontainer, a collection of deployers for loading various deployment types, and various MBean (`*-jboss-beans.xml`) and legacy MBean (`*.jboss-service.xml`) deployments. For more information refer to the *Administration and Configuration Guide*.

IMPORTANT

In this guide, we assume you have set up your environment and installed JBoss Enterprise Application Platform. If this is not the case, refer to the *Installation Guide*.

2.1. SERVER PROFILES

JBoss Enterprise Application Server, the core of JBoss Enterprise Application Platform, can be run with different basic configurations. The configuration are based on server profiles. A profile defines which services and components are available for the server instance of that profile. JBoss Enterprise Application Platform is shipped with predefined profiles; however, you can customize these profiles, remove and add individual services and components to them, as well as to create your own profiles (refer to the *Administration and Configuration Guide*).

The following JBoss Enterprise Application Server profiles are delivered with the product and located in the **<*JBOSS_HOME*>`jboss-as/server/`** directory:

all

The `all` profile covers all available services and components, including the RMI/IIOP and clustering services, which are not loaded in the default configuration.

default

The `default` profile is a base Java EE 5 server profile containing a default set of services. It covers the most frequently used services required to deploy a Java EE application. The JAXR service, the IIOP service and the clustering services are not included. This profile is applied if a server is started without specifying a configuration.

production

The `production` profile is based on the `all` profile and is intended for production. It has a reduced log verbosity, deployment scanning takes place every 60 seconds, and memory usage is tuned to accommodate production deployment requirements.

minimal

The `minimal` profile uses the minimum services required to start JBoss. It starts the logging service, a JNDI server and a URL deployment scanner to find new deployments. You can use it when using JMX/JBoss to start your own services without any other Java EE 5 technologies. It starts only the server: no web container, EJB, or JMS support is available.

standard

The **standard** profile is the JavaEE 5 certified configuration of services.

web

The **web** profile is a lightweight web-container oriented profile that previews the JavaEE 6 web profile.

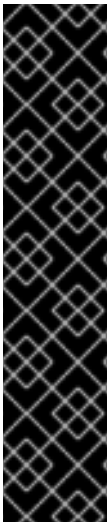
The **default** profile is used if you do not specify a configuration when running a server.

CHAPTER 3. STARTING AND STOPPING THE SERVER

The simplest way to start and stop the server is to run the server instance using the run.sh or run.bat script and run the instance as a program in foreground. For information on other ways of running the server refer to the *Administration and Configuration Guide*

3.1. STARTING THE SERVER

To start the JBoss Enterprise Application Server as a process in foreground, move to **<*JBOSS_HOME*>/jboss-as/bin** directory and execute the **run.sh** (on Red Hat Enterprise Linux) or **run.bat** (on Microsoft Windows) script. The server instance is run with the configuration set in the **default** profile as no other profile was specified.



IMPORTANT

JBoss Enterprise Application Platform binds its services to localhost (127.0.0.1) by default, instead of binding to all available interfaces (0.0.0.0). This is to prevent remote access before a server is configured and secured correctly. To enable remote access by binding JBoss services to a particular interface, run JBoss with the **-b** option. To bind to all available interfaces and re-enable the legacy behavior use **./run.sh -b 0.0.0.0** on Linux . Make sure you secure your server properly.

Using **-b** as part of the server's command line is equivalent to setting these individual properties: -Djboss.bind.address, -Djava.rmi.server.hostname, -Djgroups.bind_addr and -Dbind.address. Passing -Djboss.bind.address to the Java process as part of the **JAVA_OPTS** variable in the run scripts will not work as it is a JBoss property not a JVM property.

For more information including setting up multiple server instances on one machine and hosting multiple domains with JBoss, please refer to the *Administration and Configuration Guide*

On starting your server, your screen output should look like the following (accounting for the highlighted installation directory differences) and contain no error or exception messages:

```
[user@mypc bin]$ ./run.sh
=========================================================================
  JBoss Bootstrap Environment
  JBOSS_HOME: <JBOSS_HOME>/jboss-as
  JAVA: java
  JAVA_OPTS: -Dprogram.name=run.sh -server -Xms1503m -Xmx1503m -
Dsun.rmi.dgc.client.
gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -
Djava.net.preferIPv4Stack=true
  CLASSPATH: <JBOSS_HOME>/jboss-as/bin/run.jar
=========================================================================
```

More options for the JBoss Enterprise Application Platform **run** script are discussed in Section 3.1.1, “Starting the Server with Alternate Configuration” below.

NOTE

There is no *Server Started* message shown at the console when the server is started using the **production** profile. This message may be observed in the **server.log** file located in the **server/production/log** subdirectory.

3.1.1. Starting the Server with Alternate Configuration

Using **run.sh** without any arguments starts the server using the **default** server profile file set. To start with an alternate profile file set, pass the name of the server configuration file set (same as the name of the server configuration directory under ***<JBOSS_HOME>*/jboss-as/server/*<PROFILE>*/**) that you want to use, as the value to the **-c** command line option. For example, to start with the **minimal** profile file set you should specify:

```
[bin]$ ./run.sh -c minimal
  ...
  ...
  ...
  15:05:40,301 INFO  [Server] JBoss (MX MicroKernel) [5.0.0 (build:
SVNTag=JBoss_5_0_0 date=200801092200)] Started in 5s:75ms
```

3.2. STOPPING THE SERVER

To shut down the server, issue a **Ctrl+C** sequence in the console in which JBoss was started. Alternatively, you can use the **shutdown.sh** command.

```
[bin]$ ./shutdown.sh -S
```

NOTE

The **shutdown** command can be only used for servers that contain the **jmx-invoker-service.xml** service. Hence you cannot use the **shutdown** command with the **minimal** profile.

CHAPTER 4. SERVER AS A SERVICE

On Red Hat Enterprise Linux, Windows, and UNIX systems, you can run the JBoss Enterprise Application Platform as a service.

4.1. RUNNING AS A SERVICE ON MICROSOFT WINDOWS

1. **Download the Native Components archive**
   Navigate to http://access.redhat.com and download the Native Components archive which matches the host's architecture.

   Extract the **native** directory (and any sub-directories) contained in the archive to ***$JBOSS_HOME***.

2. **Open a command prompt with elevated privileges.**
   Navigate to **C:\Windows\System32** and right-click on **cmd.exe**. Select **Run as Administrator**.

3. **Change to the JBoss Enterprise Application Platform directory where the service installation script is located.**
   cd *$JBOSS_HOME*\native\sbin

4. **Edit service.bat to specify the profile and local IP address to be used.**
   The **service.bat** file by default uses the **default** profile and binds to IP address **127.0.0.1**, neither of which are suitable for production use.

   Change **SVCPROFILE=default**, replacing **default** with the required profile's name.

   Change **set SVCBINDIP=127.0.0.1**, replacing **127.0.0.1** with the required IP address.

   NOTE

   For a full list of profiles and the services they include, refer to the *Standard Server Profiles* section of the *Administration and Configuration Guide*.

5. **Run the service installation script.**
   service.bat install

6. **Check that the service is installed.**
   Under the Windows services list you will find this listed by the short name **JBAS52SVC** and the long name **JBoss Application Server 5.2**.

4.1.1. Removing the Service

To remove the service, do the following:

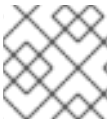
1. **Stop the service.**
   Stop the service using Service Manager.

2. **Delete the service.**
   Issue the following command from the command prompt with elevated privileges: **sc delete JBAS52SVC**

4.2. RUNNING AS A SERVICE ON RED HAT ENTERPRISE LINUX

If JBoss Enterprise Application Platform is installed using the *RPM Installation via the Red Hat Network* method, it is installed as a service. This is done by adding a new start script: **`/etc/init.d/jbossas`**, which is run automatically when Red Hat Enterprise Linux starts.

The profile used by the service is configured in **`/etc/sysconfig/jbossas`**. If you want a profile other than "default" used, change the "JBOSSCONF=" line, specifying the required profile. The service must be restarted for this change to take effect.



NOTE

Make sure that the line is not commented out (remove the # character if present).

NOTE

For a full list of profiles and the services they include, refer to the *Administration and Configuration Guide*.

CHAPTER 5. BASIC CONFIGURATION CHANGES

5.1. SECURITY CONFIGURATION

When installed from the zip archive, authentication is required to access the majority of JBoss services, including administrative services. Consoles are secured by the JAAS security domain "jmx-console". At installation, this security domain has no user accounts. This is to eliminate the possibility of default username/password based attacks. Refer to Procedure 5.1, “Create jmx-console, admin-console, and http invoker user account” to create a user account to access the consoles.

When installed via the graphical installer, a JAAS security domain and a user account is created as part of the install process. Even if you change the name of the JAAS security domain during installation, the users are stored in the same place. Follow the instructions in Procedure 5.1, “Create jmx-console, admin-console, and http invoker user account” to edit your user account, or create a new one.

IMPORTANT

This section describes only basic security configuration. This configuration is not sufficient for use in production environment. Refer to the *Security Guide* for comprehensive information on how to configure security in JBoss Enterprise Application Platform 5.

IMPORTANT

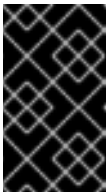
The authentication system applied to the JMX Console, Admin Console and Web Console does not block brute-force password attacks. It is recommended that in production environments, JBoss servers are protected by firewalls or reverse proxies that include measures to mitigate brute force attacks.

5.1.1. Security Configuration: JMX Console, Admin Console, HttpInvoker

Procedure 5.1. Create jmx-console, admin-console, and http invoker user account

This procedure creates user with access permissions to the admin and jmx consoles, and the http invoker

1. **Create a user in the default JAAS security domain**
   a. Edit the file **`$JBOSS_HOME/server/$PROFILE/conf/props/jmx-console-users.properties`**.
   b. Create a *username* = *password* pair.



IMPORTANT

The commented **`admin=admin`** username and password pair is an example of the username/password definition syntax. Do not use this for your user account.

2. **Grant permissions to user**
   a. Edit the file **`$JBOSS_HOME/server/$PROFILE/conf/props/jmx-console-roles.properties`**.

b. Create an entry for the user of the form:

```
username=JBossAdmin,HttpInvoker
```

JBossAdmin

Grant the user permission to access the JMX Console and Admin Console.

HttpInvoker

Grant the user permission to access the httpinvoker

5.1.2. Securing the HTTPInvoker

The HTTP Invoker is a service that provides HTTP and Remote Method Invocation (RMI) access for EJBs and the JNDI Naming service. Secure this service to prevent unauthorized access.

Procedure 5.2. Secure the HTTP Invoker

1. Edit the **<*JBOSS_HOME*>/server/<*PROFILE*>/conf/bindingservice.beans/META-INF/bindings-jboss-beans.xml** file.

2. Add the ***hostName*** and ***fixedHostName*** properties to the *deploy/legacy-invokers-service.xml* section:

```
<!-- ************ deploy/legacy-invokers-service.xml ************ --
>

<!-- RMI/JRMP invoker -->
<bean class="org.jboss.services.binding.ServiceBindingMetadata">
<property
name="serviceName">jboss:service=invoker,type=jrmp</property>
<property name="port">4444</property>
<property name="description">Socket for the legacy RMI/JRMP
invoker</property>
<property name="hostName">localhost</property>
<property name="fixedHostName">true</property>
</bean>

<!-- Pooled invoker -->
<bean class="org.jboss.services.binding.ServiceBindingMetadata">
<property
name="serviceName">jboss:service=invoker,type=pooled</property>
<property name="port">4445</property>
<property name="description">Socket for the legacy Pooled
invoker</property>
<property name="hostName">localhost</property>
<property name="fixedHostName">true</property>
</bean>
```

5.1.3. Security Configuration: Web Console

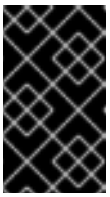
Procedure 5.3. Creating Web Console User Account

This procedure creates a user with access permissions to the web console

1. **Create a user in the web-console JAAS security domain.**

   a. Edit the file **web-console-users.properties** in **jboss-as/server/$PROFILE/deploy/management/console-mgr.sar/web-console.war/WEB-INF/classes/**.

   b. Create a *username* = *password* pair.



IMPORTANT

The commented **admin=admin** username and password is an example of the username/password definition syntax. Do not use this for your user account.

2. **Grant permissions to the user.**

   a. Edit the file **web-console-roles.properties** in **jboss-as/server/$PROFILE/deploy/management/console-mgr.sar/web-console.war/WEB-INF/classes/**.

   b. Create an entry for the user of the form:

```
username=JBossAdmin,HttpInvoker
```

JBossAdmin

Grant the user permission to access the Web-Console

HttpInvoker

Grant the user permission to access the HTTP Invoker

5.1.4. Security Configuration of JBoss Messaging

JBoss Messaging makes internal connections between nodes in order to redistribute messages between clustered destinations. These connections are made with the user name of a special reserved user whose password is specified in the property **suckerPassword** in the messaging and server configuration files.

The suckerPassword used by JBoss Messaging in a clustered environment is contained in the **jboss-as/server/$PROFILE/deploy/messaging/messaging-jboss-beans.xml** file and the **messaging-service.xml** file. These files contain directives that specify the encrypted **suckerPassword**.

Changing the Password in `messaging-jboss-beans.xml`

Complete this task to change the distribution placeholder password in **messaging-jboss-beans.xml**.

Procedure 5.4. Setting suckerPassword for JBoss Messaging

1. Navigate to the **<*JBOSS_HOME*>/server/<*PROFILE*>/deploy/messaging/** directory.

2. Open the **messaging-jboss-beans.xml** file in your preferred text editor.
3. Change the ***suckerPassword*** placeholder value from **"CHANGE ME!!"** to a plain text password:

```
<property name="suckerPassword">CHANGE ME!!</property>
```

   Make note of the new password; it will be used in the next task.

4. Save the file.

Creating the encrypted JBoss Messaging suckerPassword

Complete this task to create an encrypted suckerPassword using the JBoss Messaging SecurityUtil tool.

1. In a terminal, change to **<*JBOSS_HOME*>/server/<*PROFILE*>/deploy/messaging/**.
2. Run the following command:

```
/path/to/java/executable -cp JBOSS_HOME/client/jboss-messaging-
client.jar org.jboss.messaging.util.SecurityUtil PLAIN_TEXT_PASSWORD
```

3. *PLAIN_TEXT_PASSWORD* is the password you set in **messaging-jboss-beans.xml** in the previous task.

   As an example:

   Example 5.1. Test Encrypted Password

   Running the following command (from the ***JBOSS_HOME*/jboss-as/server/$PROFILE/deploy/messaging/** directory) ...

```
/usr/bin/java -cp ../../../../client/jboss-messaging-client.jar
org.jboss.messaging.util.SecurityUtil test
```

   ...produced the following encrypted password:

```
key len: 14 length max: 2147483647
Encoded password: 5e2c1ae5a618317
```

4. Make note of the encrypted password output; it will be used in the next task.

Specifying an encrypted suckerPassword for JBoss Messaging

Complete this task to add an encrypted suckerPassword value to JBoss Messaging configuration files.

Prerequisites

- Creating the encrypted JBoss Messaging suckerPassword

- You have a terminal open at the **<*JBOSS_HOME*>/server/<*PROFILE*>/deploy/messaging/** directory.

1. In a text editor, open the **messaging-service.xml** file.
2. Paste the encrypted password from the previous procedure into the ***SuckerPassword*** attribute:

```
<attribute name="SuckerPassword">ENCRYPTED_PASSWORD</attribute>
```

3. Save the **messaging-service.xml** file.

5.2. MEMORY SETTINGS FOR THE ENTERPRISE APPLICATION PLATFORM

The optimal memory settings for an application server depend on the deployed applications, the number of users, the virtual or physical host upon which the installation resides, and services running on the host.

The Enterprise Application Platform ships with the following default values for initial and maximum heap allocations by the JVM:

- **-Xms1303m**: initial heap size in megabytes
- **-Xmx1303m**: maximum heap size in megabytes

Follow the guidelines below when defining memory settings:

- Allocate the same values for initial and maximum heap sizes.
- Use values smaller than the host's allocatable memory.
- Be aware of other services and applications running on the host, and allow for their usage of memory.

Fine tuning the memory settings beyond these guidelines requires production-like load testing and analysis of memory usage logs, and can vary dramatically between installations and applications used with the JBoss Enterprise Application Platform.

Procedure 5.5. Changing Memory Settings for the Enterprise Application Platform on Linux

1. Navigate to ***$JBOSS_HOME*/jboss-as/bin**.
2. Open **run.conf** for editing.
3. Locate the line with the memory options:

```
JAVA_OPTS="-Xms1303m -Xmx1303m -XX:MaxPermSize=256m -
Dorg.jboss.resolver.warning=true -
Dsun.rmi.dgc.client.gcInterval=3600000 -
Dsun.rmi.dgc.server.gcInterval=3600000 -
Dsun.lang.ClassLoader.allowArraySyntax=true"
```

4. Edit the line to include the new initial and maximum heap sizes for the JVM:

```
JAVA_OPTS="-Xms INITIAL_HEAP_SIZE -Xmx MAX_HEAP_SIZE -
XX:MaxPermSize=256m -Dorg.jboss.resolver.warning=true -
Dsun.rmi.dgc.client.gcInterval=3600000 -
Dsun.rmi.dgc.server.gcInterval=3600000 -
Dsun.lang.ClassLoader.allowArraySyntax=true"
```

5. If the server is runnning, restart it to apply the new settings.

Procedure 5.6. Changing Memory Settings for the Enterprise Application Platform on Windows

1. Navigate to ***$JBOSS_HOME*/jboss-as/bin**.
2. Open **run.conf.bat** for editing.
3. Locate the line with the memory options:

```
set "JAVA_OPTS=-Xms1303m -Xmx1303m -XX:MaxPermSize=256m -
Dorg.jboss.resolver.warning=true
-Dsun.rmi.dgc.client.gcInterval=3600000 -
Dsun.rmi.dgc.server.gcInterval=3600000
-Dsun.lang.ClassLoader.allowArraySyntax=true"
```

4. Edit the line to include the new initial and maximum heap sizes for the JVM:

```
set "JAVA_OPTS=-Xms INITIAL_HEAP_SIZE m -Xmx MAX_HEAP_SIZE m -
XX:MaxPermSize=256m -Dorg.jboss.resolver.warning=true -
Dsun.rmi.dgc.client.gcInterval=3600000 -
Dsun.rmi.dgc.server.gcInterval=3600000 -
Dsun.lang.ClassLoader.allowArraySyntax=true"
```

5. If the server is runnning, restart it to apply the new settings.

5.3. SETTING THE DEFAULT SERVER APPLICATION

JBoss Enterprise Application Platform, by default, configures **<*JBOSS_HOME*>/jboss-as/server/<*PROFILE*>/deploy/ROOT.war** as the default application on the server. So accessing **http://localhost:8080/** results in displaying the index page of this application. If you want your application to be available as the default application, then you will wish to follow these steps:

- Rename **ROOT.war** in **<*JBOSS_HOME*>/jboss-as/server/<*PROFILE*>/deploy** to something else, for example, **jboss.war**.
- In your WAR file (the one which you want to be the default application), add a **jboss-web.xml**, in the **WEB-INF** folder, with a configuration for the context-root:

```
<?xml version="1.0"?>
<!DOCTYPE jboss-web PUBLIC "-//JBoss//DTD Web Application 5.0//EN"
"http://www.jboss.org/j2ee/dtd/jboss-web_5_0.dtd">
<jboss-web>
<context-root>/</context-root>
<!-- Other configurations as needed -->
</jboss-web>
```

By setting the context-root to / you are making your application the default application. Your application will now be available at **http://localhost:8080/**.

NOTE

Renaming the **ROOT.war** to **jboss.war** will make that application be available at **http://localhost:8080/jboss**

5.4. CONFIGURING LEGACY CORE SERVICES

The legacy core services specified in the ***$JBOSS_HOME***/server/***PROFILE***/conf/jboss-service.xml file are started just after server starts up the microcontainer. The file contains MBeans for various services including logging, security, JNDI, JNDIView etc. If you commenting out a service, it is not loaded on startup.

NOTE

This file will be dropped as the services are converted to microcontainer Beans or MBeans that are deployed as deploy directory services.

Note that if an MBeans definition has nested comments, you need to comment out the MBean in two sections, leaving the original comment as it was.

Example 5.2. JNDIView MBean Commented out

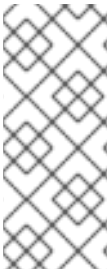
```
<!-- Section 1 commented out
<mbean code="org.jboss.naming.JNDIView"
name="jboss:service=JNDIView"
xmbean-dd="resource:xmdesc/JNDIView-xmbean.xml">
-->
<!-- The HANamingService service name -->
<!-- Section two commented out
<attribute name="HANamingService">jboss:service=HAJNDI</attribute>
</mbean>
-->
```

If you then restart JBoss, you will see that the **JNDIView** service no longer appears in the JMX Management Console (JMX Console) listing. In practice, you should rarely, if ever, need to modify this file, though there is nothing to stop you adding extra MBean entries in here if you want to. The alternative is to use a separate file in the **deploy** directory, which allows your service to be hot deployable.

5.5. CONFIGURING LOGGING SERVICE

In JBoss **log4j** is used for logging. If you are not familiar with the **log4j** package, more information is available the Jakarta web site.

Logging is controlled from the central ***$JBOSS_HOME***/server/***PROFILE***/conf/jboss-log4j.xml file. This file defines a set of appenders that specify the log files, the categories of messages the log file should contains, the message format and the level of filtering. By default, JBoss produces output to both the console and a log file (**log/server.log**).



NOTE

There are six basic log levels in log4j: **TRACE**, **DEBUG**, **INFO**, **WARN**, **ERROR** and **FATAL**. The logging threshold on the console is **INFO**, so that any informational messages, warning messages and error messages are returned to the console, while general debug and trace messages are only available in the log file. If no logging level is set for the **server.log** file, it defaults to **DEBUG**.

Always when demanding debug information, check the **server.log** file to see if there are any debug messages which might help you to track down the problem. However, be aware that just because the logging threshold allows debug messages to be displayed, that does not mean that all of JBoss produces detailed debug information for the log file. You should also consider increasing the logging limits set for individual categories.

Example 5.3. Example Log Limits

```
<!-- Limit JBoss categories to INFO -->
<category name="org.jboss">
<priority value="INFO"/>
</category>
```

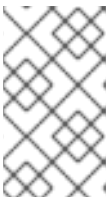
In Example 5.3, "Example Log Limits", the level of logging is set to **INFO** for all JBoss classes, with the exception of classes with more specific overrides. By default the root logger in the **jboss-log4j.xml** is set to **INFO**: any **TRACE** or **DEBUG** logger from any logger categories is not logged in any files or the console appenders. This setting is controlled through the jboss.server.log.threshold property. By default this is **INFO**. If changed to **DEBUG**, more detailed logging output is produced. Yuo can change this as follows:

- Pass the -Djboss.server.log.threshold=DEBUG parameter when starting the server:

```
./run.sh -Djboss.server.log.threshold=DEBUG
```

- Edit the **<*JBOSS_HOME*>/jboss-as/server/<*PROFILE*>/conf/jboss-log4j.xml** file directly:

```
<root>
<!-- Let us comment this out to set our own value
<priority value="${jboss.server.log.threshold}"/>-->
<priority value="DEBUG"/>
<appender-ref ref="CONSOLE"/>
<appender-ref ref="FILE"/>
</root>
```



NOTE

The **<*JBOSS_HOME*>/jboss-as/server/<*PROFILE*>/conf/jboss-log4j.xml** is scanned every 60 seconds (by default) on any changes. Therefore changing this file does not require a server restart.

As another example, let's say you wanted to set the output from the container-managed persistence engine to **DEBUG** level and to redirect it to a separate file, **cmp.log**, in order to analyze the generated SQL commands. You would add the following code to the **conf/jboss-log4j.xml** file:

```
<appender name="CMP"
class="org.jboss.logging.appender.RollingFileAppender">
<errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
<param name="File" value="${jboss.server.home.dir}/log/cmp.log"/>
<param name="Append" value="false"/>
<param name="MaxFileSize" value="500KB"/>
<param name="MaxBackupIndex" value="1"/>
<layout class="org.apache.log4j.PatternLayout">
  <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
</layout>
</appender>
<category name="org.jboss.ejb.plugins.cmp">
<priority value="DEBUG" />
<appender-ref ref="CMP"/>
</category>
```

This creates a new file appender and specifies that it should be used by the logger (or category) for the package **org.jboss.ejb.plugins.cmp**.

The file appender is set up to produce a new log file every day rather than producing a new one every time you restart the server or writing to a single file indefinitely. The current log file is **cmp.log**. Older files have the date they were written added to their file names. Please note that the **log** directory also contains HTTP request logs which are produced by the web container.

By default the **server.log** appender is configured to retain log messages between server restarts. This is controlled by the Append property on the **FILE** appender which corresponds to the **server.log** file. By default this property is set to true; if you want the **server.log** contents to be wiped out on server restarts then you can edit the ***<JBOSS_HOME>*/jboss-as/server/*<PROFILE>*/conf/jboss-log4j.xml** file to set this property value to false. For example:

```
<appender name="FILE"
class="org.jboss.logging.appender.DailyRollingFileAppender">
<errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
<param name="File" value="${jboss.server.log.dir}/server.log"/>
<param name="Append" value="false"/>
...
```

5.6. ADDITIONAL SERVICES

The non-core, hot-deployable services are added to the **deploy** directory (refer to Chapter 6, *Hot Deployment of Services*). They can be either XML descriptor files, ***-service.xml**, ***-jboss-beans.xml**, MC **.beans** archive, or JBoss Service Archive (SAR) files. SARs contains an **META-INF/jboss-service.xml** descriptor and additional resources the service requires (for example, classes, library JAR files or other archives), all packaged up into a single archive. Similarly, a **.beans** archive contains a **META-INF/jboss-beans.xml** and additional resources.

Detailed information on all these services can be found in the Administration and Configuration Guide, which also provides comprehensive information on server internals and the implementation of services, such as JTA and the J2EE Connector Architecture (JCA).

CHAPTER 6. HOT DEPLOYMENT OF SERVICES

Hot-deployable services are services which can be added to or removed from the running server. Such services are placed in the **JBOSS_DIST/jboss-as/server/<instance-name >/deploy** directory. Let's have a look at a practical example of hot deployment of services.

Make sure EAP is running and change to the **JBOSS_DIST/jboss-as/server/default/deploy** directory. Remove the **mail-service.xml** file and watch the output from the server:

```
13:10:05,235 INFO  [MailService] Mail service 'java:/Mail' removed from
JNDI

Then replace the file and watch JBoss re-install the service:

13:58:54,331 INFO  [MailService] Mail Service bound to java:/Mail
```

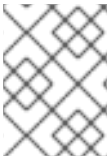
You have just undeployed the mail service during server runtime: this is hot-deployment in action.

6.1. HOT-DEPLOYMENT CONFIGURATION

Hot deployment of services in the server is controlled by the **HDScanner** MC bean configured in ***<JBOSS_HOME>*/jboss-as/server/*<PROFILE>*/deploy/hdscanner-jboss-beans.xml** file. For example, the bean sets the scanPeriod attribute, which controls the run interval for the thread that picks up the hot deployable changes. The scanPeriod property is set to 5 seconds by default (refer to Figure 6.1, "HDScanner Bean Default Configuration").

```
<bean name="HDScanner"
class="org.jboss.system.server.profileservice.hotdeploy.HDScanner">
   <property name="deployer"><inject
bean="ProfileServiceDeployer"/></property>
   <property name="profileService"><inject
bean="ProfileService"/></property>
   <property name="scanPeriod">5000</property>
   <property name="scanThreadName">HDScanner</property>
</bean>
```

Figure 6.1. HDScanner Bean Default Configuration



NOTE

Changes to the **hdscanner-jboss-beans.xml** file are hot deployable: no server restart is needed.

6.2. ADDING A CUSTOM DEPLOY DIRECTORY

EAP by default looks for deployments under the ***JBOSS_DIST*/jboss-as/server/<instance-name>/deploy** directory. However you can configure the server to include custom directories for scanning deployments. To do so, configure the **BootstrapProfileFactory** MC bean in ***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/bootstrap/profile.xml** file: add the custom directory to the applicationURIs property of the **BootstrapProfileFactory**, which defines a list of URLs scanned for applications (refer to Example 6.1, "The /home/me/myapps Directory Defined as Custom Deploy Directory").

Example 6.1. The /home/me/myapps Directory Defined as Custom Deploy Directory

```
<bean name="BootstrapProfileFactory"
class="org.jboss.system.server.profileservice.repository.
StaticProfileFactory">
   ...
   <property name="applicationURIs">
      <list elementClass="java.net.URI">
         <value>${jboss.server.home.url}deploy</value>
         <value>file:///home/me/myapps</value>
      </list>
   ...
```

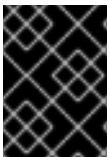
IMPORTANT

After modifying the ***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/bootstrap/profile.xml** file, you need to restart the server for the changes to take effect.

For performance reasons, adding a new deployment directory to the BootstrapProfileFactory also requires the same URL to be added to the **VFSCache** MC bean configuration in ***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/bootstrap/vfs.xml** (refer to Example 6.2, “The /home/me/myapps Directory Added to VFSCache”).

Example 6.2. The /home/me/myapps Directory Added to VFSCache

```
<bean name="VFSCache">
   ...
   <property name="permanentRoots">
      <map keyClass="java.net.URL"
valueClass="org.jboss.virtual.spi.ExceptionHandler">
         ...
         <entry>
            <key>file:///home/me/myapps</key>
            <value><inject bean="VfsNamesExceptionHandler"/></value>
         </entry>
      </map>
   </property>
   ...
```



IMPORTANT

Not adding the custom deployment directory to the **VFSCache** bean might result in growing disk space usage by the server.

CHAPTER 7. CONSOLE PAGES

Once you have started the JBoss Enterprise Application Server, you can access the *Welcome to JBoss EAP* page. This page is located at http://localhost:8080/ (or http://127.0.0.1:8080/) unless you have configured the server to bind to a different port.

From this page you can use the provided links to navigate to various console pages which allow you to manage the server:

- **`Admin console`**
- **`JMX Console`**
- **`JBoss Web Console`**
- **`Tomcat status (full) (XML)`**

By default, the console pages are secured and prompt you for a username and password when you access them. If you installed JBoss Enterprise Application Platform using the graphical installer and you want to access the JMX console, you can use the username and password you provided during installation.

If you installed using other modes (such as .zip), refer to Section 5.1, “Security Configuration”.

7.1. ADMIN CONSOLE

The Admin Console was created using Seam. It provides a hierarchical view of the available Servers, Applications and Resources associated with a single server instance of JBoss. It is a very lightweight implementation that provides most of the core functionality required by a JBoss administrator.

While it is intended to be the primary interface for the application server, not all services can currently be managed using the Admin Console. The JMX Console remains active for these services.

The Admin Console allows you to start, stop or restart the server remotely. It also allows you to view deployed web applications and more.

The Admin console is described in detail in the Admin Console User Guide.

7.2. JMX CONSOLE

Similarly to the Admin Console, the JMX Console provides access to management activities and performance data of the server. The console is accessible at http://localhost:8080/jmx-console.

The JMX Console provides a raw view of the JMX MBeans which make up the server. They can provide a lot of information about the running server and allow you to modify its configuration, start and stop components and so on.

From the JMX Console you can do the following:

- display the JNDI tree;
- generate a thread dump;
- display the memory pool usage;
- manage the deployment scanner;

- redeploy applications;
- shut down the server.

Example 7.1. JMX Console Usage

Procedure 7.1. View the JNDI Service

1. Open the JMX Console.
2. Find the **`service=JNDIView`** link and click on it.

   This particular MBean provides a service to allow you to view the structure of the JNDI namespaces within the server.
3. Now find the operation called **`list`** near the bottom of the MBean view page and click the **`invoke`** button.

   The operation returns a view of the current names bound into the JNDI tree, which is very useful when you start deploying your own applications and want to know why you can't resolve a particular EJB name.

Look at some of the other MBeans and their listed operations; try changing some of the configuration attributes and see what happens. With a very few exceptions, none of the changes made through the console are persistent. The original configuration will be reloaded when you restart JBoss, so you can experiment freely without doing any permanent damage.

7.3. JBOSS WEB CONSOLE

The JBoss Enterprise Application Platform web console (http://localhost:8080/web-console/) provides you with monitoring tools starting with the JVM Hardware environment statistics on the default page and access to monitoring tools and snapshots.

IMPORTANT

JBoss Web Console is deprecated and will be replaced with the Administration Console in the next major release.

7.4. TOMCAT STATUS

Tomcat is the default servlet container used by the JBoss Application Server. The embedded Tomcat supplies the JSP and servlet support for JBossAS.

The Tomcat status page (http://localhost:8080/status) show numerous data points relating to the JVM, thread activity and servlet status.

CHAPTER 8. DATASOURCES

JBoss Enterprise Application Platform is shipped with an example datasource configured to allow it to operate out of the box with the included Hypersonic database. The datasource is bound to the JNDI name `java:/DefaultDS` and its descriptor is the `<$JBOSS_HOME>/jboss-as/server/<PROFILE>/deploy/hsqldb-ds.xml` file.

The `DefaultDS` JNDI name and `hsqldb-ds.xml` configuration is not required for normal platform operation. Delete this datasource before deploying a production instance.

Accessing the Database Manager of the Default Hypersonic Database

To access and edit the Hypersonic database that is available by default, do the following:

1. Make sure JBoss Enterprise Application Platform is running.
2. Open the JMX console located by default on http://localhost:8080/jmx-console/.
3. On the displayed JMX Console page, click the `database=localDB,service=Hypersonic` entry in the `jboss` section.
4. On the `JMX MBean View` page of the Hypersonic MBean, click `Invoke` in the `startDatabaseManager` row to invoke the GUI interface of the database manager.

The **HSQL Database Manager** window with the default local Hypersonic database appears.

WARNING

The default persistence configuration is intended only for development purposes and to allow JBoss Enterprise Application Platform to run out of the box. However, **Hypersonic is not supported in production and should not be used in a production environment.**

Known issues with the Hypersonic database include the following:

- no transaction isolation
- thread and socket leaks (`connection.close()` does not tidy up resources)
- persistence quality (logs commonly become corrupted after a failure, preventing automatic recovery)
- database corruption
- stability under load (database processes cease when dealing with too much data)
- not viable in clustered environments.

8.1. CONFIGURING DATASOURCES

When configuring a custom datasource, you need to create the datasource configuration file, so that the JCA deployer detects and deploys the datasource: a datasource configuration file name has the **-ds.xml** suffix. The **docs/examples/jca** directory contains example files for several databases and it is recommended to adapt one of these files to create your configuration file.

NOTE

The full description of the configuration format is available in the **docs/dtd/jboss-ds_1_5.dtd** file. Additional documentation on the files and the JBoss JCA implementation are available in the *Administration and Configuration Guide*

Local transaction datasources are configured using the **local-tx-datasource** element and XA-compliant ones using **xa-datasource**. The example file **generic-ds.xml** shows how to use both types and also some of the other elements that are available for things like connection pool configuration. Examples of both local and XA configurations are available for Oracle, DB2 and Informix.

The example files **firebird-ds.xml**, **facets-ds.xml** and **sap3-ds.xml** are defined with the **connection-factories** root element rather than **datasources**. That is because they use an alternative, more generic JCA configuration syntax used with a pre-packaged JCA resource adapter. The syntax is not specific to datasource configuration and is used, for example, in the **<*JBOSS_HOME*>/jboss-as/server/<*PROFILE*>/deploy/messaging/jms-ds.xml** file to configure the JMS resource adapter.

8.2. MYSQL AS THE DEFAULT DATASOURCE

The MySQL® database has become the most popular open source database. We will configure a datasource to this database for demonstration purposes.

In this example we are using MySQL 5.1.31 and **Connector/J** 5.1.8, the official JDBC driver. Both are available at www.mysql.com.

8.2.1. Creating a Database and User

We'll assume that you have already installed MySQL, have it running, and are familiar with the basics. Run the MySQL client program from the command line so we can execute some administration commands. You should make sure that you are connected as a user with sufficient privileges (for example, by specifying the **-u** root option to run as the MySQL root user).

First create a database called **jboss** within MySQL for use by JBoss:

```
mysql> CREATE DATABASE jboss;
Query OK, 1 row affected (0.05 sec)
```

Then check that it has been created:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
```

```
| jboss    |
+----------+
1 rows in set (0.00 sec)
```

Next, create a user called **jboss** with 'password' as the password to access the database:

```
mysql> GRANT ALL PRIVILEGES ON jboss.* TO jboss@localhost IDENTIFIED BY
'password';
Query OK, 0 rows affected (0.06 sec)
```

Again, you can check that everything has gone smoothly:

```
mysql> select User,Host,Password from mysql.user;
+-------+-----------+------------------+
| User  | Host      | Password         |
+-------+-----------+------------------+
| root  | localhost |                  |
| root  | %         |                  |
|       | localhost |                  |
|       | %         |                  |
| jboss | localhost | 5d2e19393cc5ef67 |
+-------+-----------+------------------+
5 rows in set (0.02 sec)
```

8.2.2. Installing the JDBC Driver and Deploying the Datasource

To add MySQL database connection support to JBoss Enterprise Application Platform:

Procedure 8.1. Red Hat Enterprise Linux 6

1. Run the following command to install the connector:

   ```
   yum install mysql-connector-java
   ```

2. Run the following command to link the new connector to the JBoss Enterprise Application Platform installation:

   ```
   ln -s /usr/lib[64]/gcj/mysql-connector-java/mysql-connector-
   java<version>.jar.so jboss-as/server/common/lib/mysql-connector-
   java.jar
   ```

Procedure 8.2. Other Platforms

1. Download the MySQL Connector/J JDBC Connector from http://www.mysql.com

2. Deploy the connector:

   a. **To make the connector available to all server profiles;**
   Extract the **mysql-connector-java-<*version*>.jar** file to **<*JBOSS_HOME*>/server/common/lib**.

   b. **To make the connector available to selected profiles only;**
   Extract the file to the **lib** directory in those server profile directories.

Then create a file in the deploy directory called **mysql-ds.xml** with the following datasource configuration. Note that the database user name and password corresponds to the MySQL user that we created in the previous section:

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
   <local-tx-datasource>
      <jndi-name>DefaultDS</jndi-name>
      <connection-url>jdbc:mysql://localhost:3306/jboss</connection-url>
      <driver-class>com.mysql.jdbc.Driver</driver-class>
      <user-name>jboss</user-name>
      <password>password</password>
      <metadatda>
         <type-mapping>mySQL</type-mapping>
      </metadata>
      <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-
isolation>
   </local-tx-datasource>
</datasources>
```

To ensure that you have correctly configured the datasource in ***<JBOSS_HOME>*/jboss-as/server/*<PROFILE>*/deploy** folder, start the server and you will notice messages like these in the logs:

```
INFO  [ConnectionFactoryBindingService] Bound ConnectionManager
'jboss.jca:service=DataSourceBinding,name=MySqlDS' to JNDI name
'java:MySqlDS'
```

NOTE

Configuring other datasources is a similar process.

APPENDIX A. REVISION HISTORY

Revision 5.2.0-100.400 **2013-10-30** **Rüdiger Landmann**

Rebuild with publican 4.0.0

Revision 5.2.0-100 **Wed 23 Jan 2013** **Russell Dickenson**

Incorporated changes for JBoss Enterprise Application Platform 5.2.0 GA. For information about documentation changes to this guide, refer to *Release Notes 5.2.0*.

Revision 5.1.2-100 **Thu Dec 8 2011** **Jared Morgan**

Incorporated changes for JBoss Enterprise Application Platform 5.1.2 GA. For information about documentation changes to this guide, refer to *Release Notes 5.1.2*.

Revision 5.1.1-100 **Mon Jul 18 2011** **Jared Morgan**

Incorporated changes for JBoss Enterprise Application Platform 5.1.1 GA. For information about documentation changes to this guide, refer to *Release Notes 5.1.1*.

Revision 5.1.0-100 **Wed Sep 15 2010** **Laura Bailey**

Changed version number in line with new versioning requirements.
Revised for JBoss Enterprise Application Platform 5.1.0.GA.