



JBoss Enterprise Web Platform 5 Getting Started Guide

for Use with JBoss Enterprise Web Platform 5
Edition 5.1.1

Red Hat Documentation Group

JBoss Enterprise Web Platform 5 Getting Started Guide

for Use with JBoss Enterprise Web Platform 5
Edition 5.1.1

Red Hat Documentation Group

Legal Notice

Copyright © 2011 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

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.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

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

Table of Contents

Preface	3
1. Document Conventions	3
1.1. Typographic Conventions	3
1.2. Pull-quote Conventions	4
1.3. Notes and Warnings	5
2. Getting Help and Giving Feedback	5
2.1. Do You Need Help?	5
2.2. Give us Feedback	6
 Chapter 1. The JBoss Web Server - A Quick Tour	 7
1.1. Server Structure	7
1.2. Starting and Stopping the Server	7
1.2.1. Start the Server	7
1.2.2. Start the Server With Alternate Configuration	8
1.2.3. Using run.sh	8
1.2.4. Stopping the Server	9
1.2.5. Running as a Service under Microsoft Windows	9
1.3. The JMX Console	10
1.4. The JNDIView Service	11
1.5. Hot-deployment of services in JBoss	12
1.5.1. Hot-deployment configurations	13
1.5.2. Adding a custom deploy folder	13
1.6. Basic Configuration Issues	14
1.6.1. Setting your application as the default application on the server	14
1.6.2. Bootstrap Configuration	15
1.6.3. Legacy Core Services	15
1.6.4. Logging Service	16
1.6.5. Security Service	17
1.6.6. Additional Services	19
1.7. The Service Binding Manager	19
 Revision History	 21

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the [Liberation Fonts](#) set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

Mono-spaced Bold

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

To see the contents of the file **my_next_bestselling_novel** in your current working directory, enter the **cat my_next_bestselling_novel** command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

Press **Enter** to execute the command.

Press **Ctrl+Alt+F2** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **mono-spaced bold**. For example:

File-related classes include **filesystem** for file systems, **file** for files, and **dir** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications** → **Accessories** →

Character Map from the main menu bar. Next, choose **Search** → **Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic** or **Proportional Bold Italic

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username@domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh *john@example.com***.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount */home***.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books      Desktop  documentation  drafts  mss    photos  stuff  svn
books_tests Desktop1  downloads      images  notes  scripts svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```

package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        InitialContext iniCtx = new InitialContext();
        Object          ref    = iniCtx.lookup("EchoBean");
        EchoHome        home   = (EchoHome) ref;
        Echo            echo    = home.create();

        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
    }
}

```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.



Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Getting Help and Giving Feedback

2.1. Do You Need Help?

If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer Portal at <http://access.redhat.com>. Through the customer portal, you can:

- search or browse through a knowledgebase of technical support articles about Red Hat products.
- submit a support case to Red Hat Global Support Services (GSS).
- access other product documentation.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software and technology. You can find a list of publicly available mailing lists at <https://www.redhat.com/mailman/listinfo>. Click on the name of any mailing list to subscribe to that list or to access the list archives.

2.2. Give us Feedback

If you find a typographical error, or know how this guide can be improved, we would love to hear from you. Submit a report in Bugzilla against the product **JBoss Enterprise Application Platform 5** and the component **doc-Getting Started Guide**. The following link will take you to a pre-filled bug report for this product: <http://bugzilla.redhat.com/>.

Fill out the following template in Bugzilla's **Description** field. Be as specific as possible when describing the issue; this will help ensure that we can fix it quickly.

Document URL:

Section Number and Name:

Describe the issue:

Suggestions for improvement:

Additional information:

Be sure to give us your name so that you can receive full credit for reporting the issue.

Chapter 1. The JBoss Web Server - A Quick Tour

1.1. Server Structure

For a thorough explanation of the structure of the application server, see the Migration chapter of the Installation Guide that accompanies this release of JBoss Enterprise Web Platform.

1.2. Starting and Stopping the Server

1.2.1. Start the Server

Move to `$JBOSS_HOME/jboss-as-web/bin` directory and execute the `run.bat` (for Windows) or `run.sh` (for Linux) script, as appropriate for your operating system.

There is no **Server Started** message shown at the console when the server is started using the **production** profile. This message can be found in the `server.log` file located in the `$JBOSS_HOME/jboss-as-web/server/$PROFILE/log` subdirectory.



Remote connection to the JBoss Enterprise Web Platform server

The JBoss Enterprise Web Platform now binds its services to localhost (127.0.0.1) by default, instead of binding to all available interfaces (0.0.0.0). This was primarily done for security reasons because of concerns of users going to production without having secured their servers correctly. To enable remote access by binding JBoss services to a particular interface, simply run JBoss with the `-b` option. To bind to all available interfaces and re-enable the legacy behaviour use `./run.sh -b 0.0.0.0` on Linux or `run.bat -b 0.0.0.0` on Windows. In any case, be aware you still need to secure your server properly.

Using `-b` as part of the JBoss web 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 JBoss web server instances on one machine and hosting multiple domains with JBoss, please refer to the *Administration and Configuration Guide* for this distribution of JBoss Enterprise Web Platform.

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

```
[user@mypc bin]$ ./run.sh
```

```
=====

JBoss Bootstrap Environment

JBOSS_HOME: unzip_location/jboss-as

JAVA: java

JAVA_OPTS: -Dprogram.name=run.sh -server -Xms1503m -Xmx1503m -
Dsun.rmi.dgc.client.
gcInterval=36000000 -Dsun.rmi.dgc.server.gcInterval=36000000 -
Djava.net.preferIPv4Stack=true

CLASSPATH: unzip_location/jboss-as

=====
```

More options for the JBoss Enterprise Web Platform **run** script are discussed in [Section 1.2.2, “Start the Server With Alternate Configuration”](#).



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/\$PROFILE/log** subdirectory.

1.2.2. Start 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-web/server**) that you want to use, as the value to the **-c** command line option. For example, to start with the **production** profile file set you should specify:

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

1.2.3. Using run.sh

The **run** script supports the following options:

```
usage: run.sh [options]
-h, --help                Show help message
-V, --version              Show version information
--                        Stop processing options
-D<name>[=<value>]        Set a system property
-d, --bootdir=<dir>        Set the boot patch directory; Must be absolute or
url
-p, --patchdir=<dir>       Set the patch directory; Must be absolute or url
-c, --configuration=<name> Set the server configuration name
-B, --bootlib=<filename>   Add an extra library to the front bootclasspath
-L, --library=<filename>   Add an extra library to the loaders classpath
-C, --classpath=<url>      Add an extra url to the loaders classpath
-P, --properties=<url>     Load system properties from the given url
-b, --host=<host or ip>    Bind address for all JBoss services.
-g, --partition=<name>     HA Partition name (default=DefaultDomain)
-m, --mcast_port=<ip>      UDP multicast port; only used by JGroups
-u, --udp=<ip>             UDP multicast address
-l, --log=<log4j|jdk>      Specify the logger plugin type
```

1.2.4. Stopping the Server

To shutdown the server, you simply 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
```

The **shutdown** script supports the following options:

A JMX client to shutdown (exit or halt) a remote JBoss web server.

```
usage: shutdown [options] <operation>
```

options:

```
-h, --help                Show this help message (default)
-D<name>[=<value>]        Set a system property
--                        Stop processing options
-s, --server=<url>         Specify the JNDI URL of the remote server
-n, --serverName=<url>     Specify the JMX name of the ServerImpl
-a, --adapter=<name>       Specify JNDI name of the MBeanServerConnection to use
-u, --user=<name>          Specify the username for authentication
-p, --password=<name>      Specify the password for authentication
```

operations:

```
-S, --shutdown            Shutdown the server
-e, --exit=<code>         Force the VM to exit with a status code
-H, --halt=<code>         Force the VM to halt with a status code
```

Using the shutdown command requires a server configuration that contains the **jmx-invoker-service.xml** service. Hence you cannot use the shutdown command with the **minimal** profile.

1.2.5. Running as a Service under Microsoft Windows

The JBoss Enterprise Web Platform 5 comes with the necessary files to configure the server to run as a service under Windows. Distributed with JBoss Native, this enables the JBoss Enterprise Web Platform 5 to be run as a service on a Windows operating system. To install the service navigate to **\$JBoss_HOME/jboss-as-web/bin** and locate the **service.bat** file. Run the following command in a command prompt:

```
./service.bat install
```

This commands installs the JBoss Enterprise Web Platform as a service. Under the Windows services list you will find this listed by the short name **JBAS50SVC** and the long name **JBoss Application Server 5.1**.

Once the service is installed successfully, you can control and configure the service from the **Windows Services Manager** application. You can configure it to start automatically when the system is booted. You can even manually start and stop the service from the **Windows Services Manager**.

If you want to pass parameters to the server (for example, **-b**, **-c**) when running it as a service, you can do so by editing the **service.bat** to include these parameters:

```
call run.bat -c default -b localhost < .r.lock >> run.log 2>&1
```



Important

Add these parameters at both places where **run.bat** appears in the **service.bat** file.

1.3. The JMX Console

When the JBoss web server is running, you can get a live view of the server by going to the JMX console application at <http://localhost:8080/jmx-console>.

By default, the JMX console is secured and will prompt you for a username and password. 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 when it was installed. If you installed using other modes such as .zip, go to the **\$JBOSS_HOME/server/\$PROFILE/conf/props/directory** and uncomment the admin userid and password code within the **jmx-console-users.properties** file. You can add other users as needed. This will allow the defined users access to the JMX console using the username and password combination specified within the **jmx-console-users.properties** file.

See [Section 1.6.5, “Security Service”](#) for further information about the security service in JBoss Enterprise Application Platform.



Important

If you changed the **jmx-console-users.properties** file when the server was running, you may have to restart the server for the changes to take effect. In some cases, lazy loading can make this change live without restarting the server.

The JMX Console is the JBoss Management Console which 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.

For example, 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. 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.



Note

If you installed JBoss using the graphical installer, the JMX Console will prompt you for a username and password before you can access it. If you installed using other modes, you can still configure JMX Security manually. We will show you how to secure your console in [Section 1.6.5, “Security Service”](#).

1.4. The JNDIView Service

The JNDIView Service is enabled by default in the JBoss Enterprise Web Platform. This service is listed in the **jmx-console** (<http://localhost:8080/jmx-console>). Navigate to the **jboss:service=JNDIView** Mbean and click on that link. On the MBean operations page, you will find the **list** method. Click on the **Invoke** button adjacent to this **list** method.

The list operation will display the JNDI tree contents. The output will look something similar to this:

java: Namespace

```

+- securityManagement (class:
org.jboss.security.integration.JNDIBasedSecurityManagement)
+- comp (class: javax.naming.Main.Context)
+- TransactionPropagationContextImporter (class:
com.arjuna.ats.internal.jbossatx.jta.PropagationContextManager)
+- policyRegistration (class: org.jboss.security.plugins.JBossPolicyRegistration)
+- Mail (class: javax.mail.Session)
+- TransactionPropagationContextExporter (class:
com.arjuna.ats.internal.jbossatx.jta.PropagationContextManager)
+- ProfileService (class:
org.jboss.system.server.profileservice.repository.AbstractProfileService)
+- DefaultDS (class: org.jboss.resource.adapter.jdbc.WrapperDataSource)
+- jaas (class: javax.naming.Context)
| +- HsqlDbRealm (class: org.jboss.security.plugins.SecurityDomainContext)
+- TransactionSynchronizationRegistry (class:
com.arjuna.ats.internal.jta.transaction.arjunacore.TransactionSynchronizationRegistry
Imple)
+- SecurityProxyFactory (class: org.jboss.security.SubjectSecurityProxyFactory)
+- TransactionManager (class:
com.arjuna.ats.jbossatx.jta.TransactionManagerDelegate)
+- timedCacheFactory (class: javax.naming.Context)
Failed to lookup: timedCacheFactory, errmsg=org.jboss.util.TimedCachePolicy cannot
be cast to javax.naming.NamingEnumeration

```

Global JNDI Namespace

```

+- UserTransactionSessionFactory (proxy: $Proxy93 implements interface
org.jboss.tm.usertx.interfaces.UserTransactionSessionFactory)
+- SecureDeploymentManager (class: org.jnp.interfaces.NamingContext)
| +- remote[link -> DeploymentManager] (class: javax.naming.LinkRef)
+- SecureManagementView (class: org.jnp.interfaces.NamingContext)
| +- remote[link -> ManagementView] (class: javax.naming.LinkRef)
+- DeploymentManager (class: org.jboss.aop.generatedproxies.AOPProxy$4)
+- ProfileService (class: org.jboss.aop.generatedproxies.AOPProxy$2)
+- SecureProfileService (class: org.jnp.interfaces.NamingContext)
| +- remote[link -> ProfileService] (class: javax.naming.LinkRef)
+- UserTransaction (class: org.jboss.tm.usertx.client.ClientUserTransaction)
+- jmx (class: org.jnp.interfaces.NamingContext)
| +- invoker (class: org.jnp.interfaces.NamingContext)
| | +- RMIAdaptor (proxy: $Proxy89 implements interface
org.jboss.jmx.adaptor.rmi.RMIAdaptor, interface
org.jboss.jmx.adaptor.rmi.RMIAdaptorExt)
| | +- rmi (class: org.jnp.interfaces.NamingContext)
| | | +- RMIAdaptor[link -> jmx/invoke/RMIAdaptor] (class: javax.naming.LinkRef)
+- TomcatAuthenticators (class: java.util.Properties)
+- ManagementView (class: org.jboss.aop.generatedproxies.AOPProxy$3)

```

This details the JNDI names to which your EJBs are bound.

1.5. Hot-deployment of services in JBoss

Hot-deployable services are those which can be added to or removed from the running server. These are placed in the **\$JBOSS_HOME/jboss-as-web/server/\$PROFILE/deploy** directory. Let's have a look at a practical example of hot-deployment of services in JBoss.

Start JBoss if it isn't already running and take a look at the **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
```

This is hot-deployment in action.

1.5.1. Hot-deployment configurations

Hot deployment of services in the server is controlled by the HDScanner MC bean configured in **\$JBOSS_HOME/jboss-as-web/server/\$PROFILE/deploy/hdscanner-jboss-beans.xml** file. For the **default** server configuration the scanPeriod is set to 5 seconds:

```
<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>
```

The scanPeriod attribute controls the interval for thread which picks up the hot deployable changes.



Note

The changes to the **hdscanner-jboss-beans.xml** file itself are hot deployable. No server restart is needed.

1.5.2. Adding a custom deploy folder

JBoss web server by default looks for deployments under the **\$JBOSS_HOME/jboss-as-web/server/\$PROFILE/deploy** folder. However you can configure the server to even include your custom folder for scanning deployments. This can be done by configuring the **BootstrapProfileFactory** MC bean in **\$JBOSS_HOME/jboss-as-web/server/\$PROFILE/conf/bootstrap/profile.xml** file. The applicationURLs property of the **BootstrapProfileFactory** accepts a list of URLs which will be scanned for applications. You can add your custom deploy folder to this list. For example, if you want **/home/me/myapps** to be scanned for deployments, then you can add the following:

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


**Important**

Modifying the **\$JBOSS_HOME/jboss-as-web/server/\$PROFILE/conf/bootstrap/profile.xml** requires a server restart, for the changes to take effect.

For performance reasons, adding a new deployment folder to the **BootstrapProfileFactory** also requires the same URL to be added to the **VFSCache** MC bean configuration in **\$JBOSS_HOME/jboss-as-web/server/\$PROFILE/conf/bootstrap/vfs.xml**. For example:

```
<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>
  ...
</bean>
```

**Important**

Not adding the custom deployment folder to **VFSCache** might result in growing disk space usage by the server, over a period of time.

1.6. Basic Configuration Issues

Now that we have examined the JBoss web server, we will take a look at some of the main configuration files and what they are used for. All paths are relative to the server configuration directory (**server/default**, for example).

1.6.1. Setting your application as the default application on the server

JBoss web server by default configures **\$JBOSS_HOME/jboss-as-web/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-web/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**

1.6.2. Bootstrap Configuration

The microcontainer bootstrap configuration is described by the **conf/bootstrap.xml** and the **conf/bootstrap/*.xml** it references. It is expected that the number of bootstrap beans will be reduced in the future. It is not expected that you would need to edit the bootstrap configuration files for a typical installation.

1.6.3. Legacy Core Services

The legacy core services specified in the **conf/jboss-service.xml** file are started just after server starts up the microcontainer. If you have a look at this file in an editor you will see MBeans for various services including logging, security, JNDI, JNDIView etc. Try commenting out the entry for the **JNDIView** service.



Note

Eventually this file will be dropped as the services are converted to microcontainer beans or mbeans that are deployed as deploy directory services.

Note that because the mbeans definition had nested comments, we had to comment out the mbean in two sections, leaving the original comment as it was.

```
<!-- 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.

1.6.4. Logging Service

In JBoss **log4j** is used for logging. If you are not familiar with the **log4j** package and would like to use it in your applications, you can read more about it at the Jakarta web site (<http://jakarta.apache.org/log4j/>).

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

There are 6 basic log levels used: **TRACE**, **DEBUG**, **INFO**, **WARN**, **ERROR** and **FATAL**. The logging threshold on the console is **INFO**, which means that you will see informational messages, warning messages and error messages on the console but not general debug and trace messages. If no logging level is set for the **server.log** file, it defaults to **DEBUG**.

If things are going wrong and there doesn't seem to be any useful information in the console, always 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 doesn't mean that all of JBoss will produce detailed debug information for the log file. You will also have to boost the logging limits set for individual categories. Take the following category for example.

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

This limits the level of logging to **INFO** for all JBoss classes, apart from those which have more specific overrides provided. By default the root logger in the **jboss-log4j.xml** is set to **INFO**. This effectively means that any **TRACE** or **DEBUG** logger from any logger categories will not be logged in any files or the console appender. This setting is controlled through the `jboss.server.log.threshold` property. By default this is **INFO**. If you were to change this to **DEBUG**, it would produce much more detailed logging output. In order to change this there are two options:

- You can pass the `-Djboss.server.log.threshold=DEBUG` parameter while starting the server:

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

- You can edit the **\$JBOSS_HOME/jboss-as-web/server/\$PROFILE/conf/jboss-log4j.xml** file directly in order to set this property:

```
<root>
  <!-- Let's 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 `$JBOSS_HOME/jboss-as-web/server/$PROFILE/conf/jboss-log4j.xml` is scanned every 60 seconds (by default) to check for any changes. Changing this file does not require a server restart as the changes will be hot deployed within the next 60 seconds following the change.

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 filenames. 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 `$JBOSS_HOME/jboss-as-web/server/$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"/>
  ...
```

1.6.5. Security Service

The security domain information is stored in the file `conf/login-config.xml` as a list of named security domains, each of which specifies a number of JAAS ^[1] login modules which are used for

authentication purposes in that domain. When you want to use security in an application, you specify the name of the domain you want to use in the application's JBoss-specific deployment descriptors, **jboss.xml** (used in defining jboss specific configurations for an application) and/or **jboss-web.xml** (used in defining jboss for a Web application. We'll quickly look at how to do this to secure the JMX Console application which ships with JBoss.

Almost every aspect of the JBoss web server can be controlled through the JMX Console, so it is important to make sure that, at the very least, the application is password protected. Otherwise, any remote user could completely control your server. To protect it, we will add a security domain to cover the application. This can be done in the **jboss-web.xml** file for the JMX Console, which can be found in **deploy/jmx-console.war/WEB-INF/** directory. Uncomment the **security-domain** in that file, as shown below.

```
<jboss-web>
  <security-domain>
    java:/jaas/jmx-console
  </security-domain>
</jboss-web>
```

This links the security domain to the web application, but it does not tell the web application what security policy to enforce, what URLs are we trying to protect, and who is allowed to access them. To configure this, go to the **web.xml** file in the same directory and uncomment the **security-constraint** that is already there. This security constraint will require a valid user name and password for a user in the **JBossAdmin** group.

```
<!--
  A security constraint that restricts access to the HTML JMX console
  to users with the role JBossAdmin. Edit the roles to what you want and
  uncomment the WEB-INF/jboss-web.xml/security-domain element to enable
  secured access to the HTML JMX console.
-->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>HtmlAdaptor</web-resource-name>
    <description>
      An example security config that only allows users with the
      role JBossAdmin to access the HTML JMX console web application
    </description>
    <url-pattern>/*</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  <auth-constraint>
    <role-name>JBossAdmin</role-name>
  </auth-constraint>
</security-constraint>
```

That's great, but where do the user names and passwords come from? They come from the **jmx-console** security domain we linked the application to. We have provided the configuration for this in the **conf/login-config.xml**.

```
<application-policy name="jmx-console">
  <authentication>
    <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
flag="required">
      <module-option name="usersProperties">
        props/jmx-console-users.properties
      </module-option>
      <module-option name="rolesProperties">
        props/jmx-console-roles.properties
      </module-option>
    </login-module>
  </authentication>
</application-policy>
```

This configuration uses a simple file based security policy. The configuration files are found in the **conf/props** directory of your server configuration. The usernames and passwords are stored in the **conf/props/jmx-console-users.properties** file and take the form "**username=password**". To assign a user to the **JBossAdmin** group add "**username=JBossAdmin**" to the **jmx-console-roles.properties** file (additional roles on that username can be added comma separated). The existing file creates an **admin** user with the password **admin**. For security, please either remove the user or change the password to a stronger one.

JBoss will re-deploy the JMX Console whenever you update its **web.xml**. You can check the server console to verify that JBoss has seen your changes. If you have configured everything correctly and re-deployed the application, the next time you try to access the JMX Console, it will ask you for a name and password. [2]

The JMX Console is not the only web-based management interface in JBoss Enterprise Web Platform. The Enterprise Web Platform Administration Console is located in **deploy/admin-console.war** and shares the security domain associated with the JMX Console. For more information, refer to the *Administration Console User Guide*, available from http://www.redhat.com/docs/en-US/JBoss_Enterprise_Web_Platform/.

1.6.6. Additional Services

The non-core, hot-deployable services are added to the **deploy** directory. 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 *JBoss Enterprise Web Platform: 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).

1.7. The Service Binding Manager

JBoss web server uses various ports for the services that it provides (for example, port 8080 for HTTP, 1099 for JNDI). The Service Binding Manager (SBM) service provides a centralized location where settings for all services that need to bind to ports can be configured. SBM can be used to configure different sets of port bindings for a server instance. A system property on the SBM controls which named set (for example, ports-default, ports-01) is used by a particular server instance. If you want to run multiple server instances on the same system then you can configure the SBM on each instance to

use a different named binding set. You can even use SBM to switch to a different binding set (for example, 8180 port for HTTP instead of the default 8080) for a server instance.

In a typical configuration, the **ports-default** set uses the standard ports (for example, JNDI on port 1099), with **ports-01** increasing each port value by 100 (for example, JNDI on 1199), **ports-02** by 200 and so on.

SBM is configured through the **\$JBOSS_HOME/jboss-as-web/server/\$PROFILE/conf/bindingservice.beans/META-INF/bindings-jboss-beans.xml** file. The configuration of the **ServiceBindingManager** involves three primary elements :

- ▶ A set of beans containing standard (default) binding configuration data. These are the base values (for example, JNDI on 1099) used to drive **ports-default**, **ports-01** and so on.
- ▶ A number of beans defining **ServiceBindingSets**, for example, **ports-default**, **ports-01**, **ports-02**. The sets of standard bindings are combined with each of these, along with an offset value (for example, 100 for **ports-01**) that should be applied to the standard port values to create the binding values for that set.
- ▶ The **ServiceBindingManager** service bean itself. This has the standard bindings and the **ServiceBindingSets** injected into it. It is also configured with the name of the binding set the particular server instance should use. The name of the binding set to be used is configurable from the command line by using the system property `jboss.service.binding.set`. The default value is **ports-default**.

```
<bean name="ServiceBindingManagementObject"
class="org.jboss.services.binding.managed.ServiceBindingManagementObject">
  <constructor>

    <parameter>
      ${jboss.service.binding.set:ports-default}
    </parameter>

    ...
  </constructor>
</bean>
```

To switch to a different set of ports than the ones used by default, you can start the server by passing the `-Djboss.service.binding.set` property to the run command as follows:

```
./run.sh -Djboss.service.binding.set=ports-01
```

This will instruct the server to use the group of ports configured in the **ports-01** binding set.

[1] The Java Authentication and Authorization Service. JBoss uses JAAS to provide pluggable authentication modules. You can use the ones that are provided or write your own if you have more specific requirements.

[2] Since the username and password are session variables in the web browser you may need to restart your browser to use the login dialog window.

Revision History

Revision 5.1.1-104.400	2013-10-31	Rüdiger Landmann
Rebuild with publican 4.0.0		
Revision 5.1.1-104	2012-07-18	Anthony Towns
Rebuild for Publican 3.0		
Revision 5.1.1-100	Mon Jul 18 2011	Jared Morgan
Incorporated changes for JBoss Enterprise Web Platform 5.1.1 GA. For information about documentation changes to this guide, refer to <i>Release Notes 5.1.1</i> .		