Novell exteNd

5.2.1

www.novell.com

JBOSS MIGRATION GUIDE

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to makes changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

This product may require export authorization from the U.S. Department of Commerce prior to exporting from the U.S. or Canada.

Title to the Software and its documentation, and patents, copyrights and all other property rights applicable thereto, shall at all times remain solely and exclusively with SilverStream and its licensors, and you shall not take any action inconsistent with such title. The Software is protected by copyright laws and international treaty provisions. You shall not remove any copyright notices or other proprietary notices from the Software or its documentation, and you must reproduce such notices on all copies or extracts of the Software or its documentation. You do not acquire any rights of ownership in the Software.

Patents pending.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.

www.novell.com

Novell exteNd 5.2.1 *JBoss Migration Guide*
February 2005

Online Documentation: To access the online documemntation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

ConsoleOne, GroupWise, iChain, NetWare, and Novell are registered trademarks of Novell, Inc.

eDirectory, exteNd, exteNd Composer, exteNd Director, jBroker, Novell eGuide, and Nsure are trademarks of Novell, Inc.

SilverStream Trademarks

SilverStream is a registered trademark of SilverStream Software, LLC.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

Third-Party Software Legal Notices

The Apache Software License, Version 1.1

Autonomy

Bouncy Castle

Castor Library

5. Due credit should be given to the ExoLab? Project (http://www.exolab.org/).

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

Indiana University Extreme! Lab Software License

Version 1.1.1

Copyright (c) 2002 Extreme! Lab, Indiana University. All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Indiana University Extreme! Lab (http://www.extreme.indiana.edu/)."

Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear.

4. The names "Indiana University" and "Indiana University Extreme! Lab" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact http://www.extreme.indiana.edu/.
5. Products derived from this software may not use "Indiana University" name nor may "Indiana University" appear in their name, without prior written permission of the Indiana University.

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

JDOM.JAR

Copyright (C) 2000-2002 Brett McLaughlin & Jason Hunter. All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution.
3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@jdom.org.
4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management (pm@jdom.org).

In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (http://www.jdom.org/)."

Alternatively, the acknowledgment may be graphical using the logos available at http://www.jdom.org/images/logos.

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

Phaos

This Software is derived in part from the SSLavaTM Toolkit, which is Copyright ©1996-1998 by Phaos Technology Corporation. All Rights Reserved. Customer is prohibited from accessing the functionality of the Phaos software.

W3C

W3C® SOFTWARE NOTICE AND LICENSE

This work (and included software, documentation such as READMEs, or other related items) is being provided by the copyright holders under the following license. By obtaining, using and/or copying this work, you (the licensee) agree that you have read, understood, and will comply with the following terms and conditions.

Permission to copy, modify, and distribute this software and its documentation, with or without modification, for any purpose and without fee or

Contents

About This Book

Purpose

This book provides a quick introduction to the JBoss application server and provides instructions for migrating to this application server from the exteNd Application Server. It is intended to help users make the transition to JBoss, but does not give complete details on configuring and managing the JBoss server.

Audience

This book is intended for users who are already familiar with the exteNd Application Server.

Prerequisites

This book assumes you are familiar with the Java programming language, the Internet, and Web applications.

See Also

📖 For complete information on the JBoss application server, see your JBoss documentation.

1 Introduction to JBoss for exteNd Users

This chapter provides an introduction to the JBoss Application Server for users familiar with the exteNd Application Server. It includes these topics:

- About JBoss
- JBoss and the exteNd Application Server compared

About JBoss

The JBoss Application Server is an Open Source application server written in Java. JBoss is a complete J2EE 1.4 certified implementation, supporting the full range of J2EE services. JBoss allows you to deploy applications easily to a wide range of platforms, including Linux, Solaris, Microsoft, and Unix.

J2EE services

JBoss offers full support for these major areas of the J2EE 1.4 specification:

- J2EE Web Services and the Service Oriented Architecture (SOA)
- Java Messaging Service (JMS)
- Enterprise JavaBeans (EJB)
- Java Transaction API (JTA)
- Java Authentication and Authorization Service (JAAS)
- J2EE Connector Architecture (JCA)
- Java Authorization Contract for Containers (JACC)
- JavaServerPages and servlets (provided through integration with Tomcat)

The JBoss architecture is based on a **Java Management Extension (JMX)** infrastructure. This infrastructure makes it possible for JBoss to integrate with a variety of external software components. These components are defined as Managed Bean (MBean) services that can be loaded into JBoss. Once defined, these components can be managed through JMX.

JBoss extensions

JBoss also includes several key features that build on the J2EE architecture:

- Built-in support for the Aspect-Oriented Programming (AOP) model, which greatly improves developer productivity
- Integration with Hibernate, an object persistence framework developed by JBoss
- Enterprise-class clustering and distributed cache support

JBoss server configurations

JBoss supports several server configurations out of the box:

- The **all** configuration starts all of the services, including clustering.
- The **default** configuration starts all J2EE services using the JBoss optimized class loader. Since the components are deployed in the same Java Virtual Machine (JVM), this configuration results in the best performance. However, the deployed applications are not fully compartmentalized in this configuration.
- The **minimal** configuration starts the JBoss microkernel, as well as the JMX MBean server and the JNDI naming service.

Custom configurations You can also develop your own custom server configurations. For example, you might want to make a copy of the contents of the **all** configuration (jboss-4.0.*x*\server\all) and name the custom configuration **exteNd**. In this case, you would modify the configuration files under \jboss-4.0.*x*\server\exteNd to satisfy your own application or performance requirements.

NOTE: To run the custom exteNd configuration, you would issue the command `run - c exteNd`.

Where to find more information on JBoss

Here's a list of useful links to the JBoss documentation:

📖 For introductory material on the JBoss application server, see Getting Started with JBoss 4.0.

📖 For complete information on configuring and managing the server, see The JBoss 4 Application Server Guide.

📖 For details on what's changed in Version 4, see What's New in JBoss AS 4.0.

JBoss and the exteNd Application Server compared

This section presents a feature-by-feature comparison of JBoss and the exteNd Application Server.

	JBoss	exteNd Application Server
Architecture	Pluggable architecture, based on JMX.	Fixed architecture.
Server repository	File-based.	Requires a database.
Server configuration	Configured through a set of XML files in the JBoss file structure. These files are grouped in several configuration sets: ◆ all ◆ default ◆ minimal	Uses SMC and SilverCMD for server configuration.
Server management	Can be managed through any JMX-compliant management application. At installation time, JBoss includes the JMX Management Console for server management.	Uses SMC and SilverCmd for server management.

	JBoss	**exteNd Application Server**
Registry	Does not use the Windows registry, making it easy to maintain several installations on the same system.	Uses the Windows registry for some settings, which it's difficult to maintain several installations on the same system.
Web container	Web container is a pluggable MBean. By default, JBoss is integrated with Tomcat.	Installs a built-in Web container.
Java environment	Requires a pre-installed Java Development Kit (JDK).	Installs its own version of the Java Runtime Environment (JRE).
Classloading	Uses a flat class loading model. All classes are loaded by the same classloader. If applications cannot share classes because of version conflicts, you need to scope the classes. For more information on classloading in JBoss, see "Classloading issues" on page 46.	JARs in AGCLASSPATH variable and AgJars.conf file are loaded by the system classloader. Each application (EAR or WAR) has its own classloader.
Clustering	JBoss has clustering built in and enabled in its **all** configuration. JBoss also uses JavaGroups, which allow developers to create reliable multicast applications without making code changes.	Clustering components (Cache Manager, Load Manager, and Dispatcher) are available with certain editions of the application server. Clustering requires more work to configure.

2 Installing and Configuring JBoss

This chapter provides instructions for installing and configuring the JBoss Application Server. It includes these topics:

- Installing the JDK
- Installing JBoss
- Running the server
- Basic server configuration
- Working with databases
- Basic server management
- Where to find more information

Installing the JDK

Before you install the JBoss Application Server, you need to have an up-to-date version of the Java Development Kit (JDK) on your machine. Be sure to install the JDK and not the JRE.

NOTE: You need to have **JDK 1.4.2**.

To install the JDK:

1 Download the JDK from the Sun web site to a directory on your machine.

2 Navigate to the directory and run the Setup program.

3 Set the JAVA_HOME environment variable to point to the location of the JDK:

Platform	Instructions
Linux	Export JAVA_HOME=/usr/java/j2sdk*version*. To avoid having to set JAVA_HOME each time you open a new shell to start the server, you should add the export statement to the .bashrc file in the user home directory. For example: `export JAVA_HOME=/usr/java/j2sdk1.4.2_06`
Windows	Open **Control Panel>System**. On the **Advanced** tab, click **Environment Variables**. In the System variable section, add a variable called JAVA_HOME and specify the target directory for the JDK as the value.

4 Set the PATH environment variable to point to the bin directory of the JDK:

Platform	Instructions
Linux	To avoid having to set the PATH each time you open a new shell to start the server, you should add the export statement to the .bashrc file in the user home directory. For example: `export PATH=$PATH:$JAVA_HOME/bin`

Platform	Instructions
Windows	Open **Control Panel>System**. On the **Advanced** tab, click **Environment Variables**. In the System variable section, add a variable called PATH and specify the bin directory for the JDK as the value. `PATH = C:\j2sdk1.4.2_06\bin`

5 Verify that the JDK is working properly and make sure that the java executable is in your path.

To do this, you can execute `java -version` at a command prompt. If the JDK installation was successful, you should see output similar to the following:

```
C:\>java -version
java version "1.4.2_05"
Java(VM) 2 Runtime Environment, Standard Edition (build 1.4.2_05-b04)
Java Hotspot (TM) Client VM (build 1.4.2_05-b04, mixed mode)
```

Configuring the JDK to work with exteNd Director Before deploying an exteNd Director project to JBoss, you need to make sure that the exteNd copies of these JAR files are available to the Java runtime:

- Phaos_Crypto_FIPS.jar
- Phaos_Security_Engine.jar
- Phaos_SSLava.jar

Copy these files from this exteNd directory:

```
Common\jre\lib\ext
```

NOTE: If you are developing on a Windows platform and deploying to a server running on UNIX, copy the Phaos_Crypto_FIPS_UNIX.jar from the Common\lib\other directory to the JRE directory. Rename it Phaos_Crypto_FIPS.jar.

Installing JBoss

The JBoss Application Server is available in source and binary form. If you are new to JBoss, you should plan on using the binary form, since it runs out of the box. The binary distribution is available in these file formats:

- .zip
- .tar.gz
- .bz2

When you download the product, select the file format that is most suitable for your platform.

To install the JBoss Application Server:

1 Download the product from the JBoss web site.

2 Extract the server to a folder of your choice.

Once it's been extracted, the folder name should be `jboss-4.0.`*`x`*. The version number suffix will, of course, vary depending on which version you installed.

NOTE: Be sure not to extract JBoss to a directory that contains spaces in the path. Spaces may cause problems for the JBoss installation.

Running the server

This section provides instructions for starting and stopping the JBoss server.

Starting the server

To start the JBoss Application Server:

1 Navigate to the `bin` directory under the main JBoss directory (`jboss-4.0.x`).

2 Execute the `run` command:

On Windows, you need to execute `run.bat`. On Linux and UNIX, you need to execute `run.sh`.

When the server starts, it deploys and starts all components, logging output to the command window. If the server starts successfully, you should see output similar to the following at the end of the log:

```
23:35:58,786 INFO [Server] JBoss (MX MicroKernel) [4.0.0 (build:
CVSTag=JBoss_4_0_0 date=200409200418)]
```

Adding parameters to the run command If you issue the `run` command without any parameters, the default configuration for the server is used. To use one of the other configurations (minimal, all, or a custom configuration), you need to add the `-c` parameter. For example, to start the server with the all configuration, you would use this command:

```
run -c all
```

If you developed a custom server configuration, you would specify the name of this configuration with the `-c` parameter. For example, if you created a custom configuration called **exteNd**, you would start the server with this command:

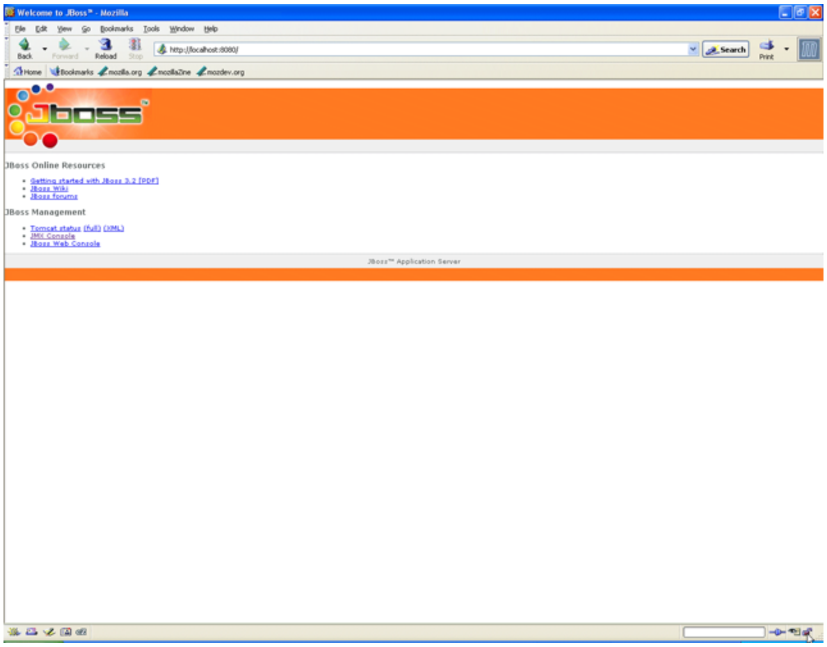
```
run -c exteNd
```

To see a listing of available start-up options, add the `-h` parameter:

```
run -h
```

Verifying that the server is working To make sure that the JBoss server is running properly, you can test the server with this URL:

```
http://localhost:8080
```



JBoss Online Resources

- Getting started with JBoss 3.2 [PDF]
- JBoss Wiki
- JBoss forums

JBoss Management

- Tomcat status (full) (XML)
- JMX Console
- JBoss Web Console

JBoss™ Application Server

NOTE: 8080 is the default port for the JBoss server. The default HTTP server URL in JBoss points to the ROOT.war in the jbossweb-tomcat50.sar.

Stopping the server

To stop the JBoss server:

1. Navigate to the `bin` directory under the main JBoss directory (`jboss-4.0.x`).
2. Execute the `shutdown` command with the `-S` parameter:

   ```
   shutdown -S
   ```

   On Windows, you need to execute `shutdown.bat`. On Linux and UNIX, you need to execute `shutdown.sh`.

You can also shutdown the server by typing Ctrl-C in a command window.

Basic server configuration

When you unpack the binary distribution of JBoss, you will see five main folders under the main JBoss directory (`jboss-4.0.x`):

Folder	Description
bin	Contains scripts for starting and stopping the server.
client	Contains JAR files and configuration files needed to run a Java client.
docs	Contains the DTD files used in JBoss, as well as sample configuration files for setting up datasources.
lib	Contains JAR files needed to run the JBoss microkernel.
server	Provides server configuration files. Each server configuration is defined in its own subdirectory (all, default, and minimal).

Quick tour of JBoss The *Getting Started with JBoss 4.0* manual includes a chapter called The JBoss Server - A Quick Tour that provides a useful introduction to the JBoss server environment. Read this chapter for more information on basic server configuration.

Working with databases

JBoss uses a set of Java Database Connectivity (JDBC) configuration files to control database access. The JBoss Application Server uses Hypersonic as its default database. To use JBoss with another kind of database, you need to modify these files:

- *datasource*-ds.xml
- Standardjaws.xml or jaws.xml
- Standardjbosscmp-jdbc.xml

Oracle Database Configuration

To configure JBoss for Oracle database access:

1 Place Oracle's driver classes in the CLASSPATH.

2 Copy Oracle's JDBC driver .jar file **/jdbc/lib/ojdbc14.jar** to the JBoss- 4.0.*x*/server/*config-folder*/lib directory, where *config-folder* is the name of the server configuration folder you want to use.

NOTE: Be sure to download the latest version of the Oracle JDBC driver from the Oracle web site, together with the free Instant Client. This version has significant performance enhancements over previous versions.

Modify the Oracle-ds.xml configuration file

Before you begin working on the datasource file, you need to determine whether you will be using Oracle’s transactional (XA) support. Once you have made this determination, you can copy one of the example files to your deploy directory, change the name to match the JNDI name for your datasource, and begin making changes, as follows:

- If you are using Oracle's transactional (XA) datasource, you can copy /docs/examples/jca/oracle-xa-ds.xml to JBoss- 4.0.*x*/server/*config-folder*/deploy directory, where *config-folder* is the name of the server configuration folder you want to use.
- To configure with the non-XA data source, copy /docs/examples/jca/oracle-ds.xml, to JBoss-4.0.*x*/server/*config-folder*/deploy directory, where *config-folder* is the name of the server configuration folder you want to use.

NOTE: The format for the file name is *datasource*-ds.xml.

The <driver-class/> and <connection-url/> settings for Oracle are presented below:

Driver	Driver class	Connection URL
Oracle OCI Type 2 Driver	oracle.jdbc.driver.OracleDriver	jdbc:oracle:oci:@*database*
Oracle OCI Thin Type 4 Driver	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@*host*:*port*:*database*
Oracle OCI XA Type 2 Driver	oracle.jdbc.xa.client.OracleXADataSource	jdbc:oracle:thin:@*host*:*port*:*database*
Oracle OCI Type 2 Driver	oracle.jdbc.driver.OracleDriver	jdbc:oracle:oci:@*database*

In the Connection URL setting, *host* is the HOST value specified in the /network/ADMIN/tnsnames.ora file, and *port* is the PORT value specified in the tnsnames.ora file, and *database* is the database name.

Modify the standardjaws.xml or jaws.xml configuration file

The **standardjaws.xml** file is the standard descriptor for mapping Container Managed Persistence (CMP) entity EJBs. If you plan to use the standard configuration for EJB processing using CMP, you need to edit this file. If you want to use a custom configuration for mapping CMP entity EJBs, you need to edit the **jaws.xml** file instead.

Whether you use standardjaws.xml or jaws.xml, you need to set the <datasource> and <type-mapping> elements, as shown below:

```
<jaws>
  <datasource>java:/OracleDS</datasource>
  <type-mapping>Oracle8</type-mapping>
</jaws>
```

Copy the file to the META-INF directory within the EJB jar file.

Modify the standardjbosscmp-jdbc.xml or jbosscmp-jdbc.xml configuration file

The **standardjbosscmp-jdbc.xml** file is the standard descriptor for configuring the JBoss CMP container. If you want to use a custom configuration for the JBoss CMP container, you can edit the **jbosscmp-jdbc.xml** file instead.

Whether you use standardjbosscmp-jdbc.xml or jbosscmp-jdbc.xml file, you need to set the <datasource> and <datasource-mapping> elements, as shown below:

```
<jbosscmp-jdbc>
  <defaults>
    <datasource>java:/OracleDS</datasource>
    <datasource-mapping>Oracle8</datasource-mapping>
  </defaults>
</jbosscmp-jdbc>
```

Copy the file to the META-INF directory within the EJB jar file.

MySQL Database Configuration

To configure JBoss for MySQL database access:

1. Place the MySQL driver classes into the CLASSPATH.
2. Copy the .jar file **\exteNd521\MySQL\jdbc\mysql-connector-java-bin.jar** to the JBoss-4.0.*x*/server/*config-folder*/lib directory, where *config-folder* is the name of the server configuration folder you want to use.

Modify the mysql-ds.xml configuration file

Before you begin working on the datasource file, you can copy the MySQL data source from /docs/examples/jca/mysql-ds.xml to the /server/*config-folder*/deploy directory, where *config-folder* is the name of the server configuration folder you want to use. Once you've copied the file, change the name to match the JNDI name for your datasource, and begin making modifications, as follows:

- Set the <driver-class/> to com.mysql.jdbc.Driver.
- Set the <connection-url/> to jdbc:mysql://*mysqlhost*/*database*, where *mysqlhost* is the MySQL host server and *database* is the MySQL database.

NOTE: The format for the file name is *datasource*-ds.xml.

Here's an example that shows what the file would contain for the ExpressPortal database:

```
<datasources>
  <local-tx-datasource>
    <jndi-name>ExpressPortal</jndi-name>
    <connection-url>jdbc:mysql://localhost:63306/expressportal</connection-url>
    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <user-name>root</user-name>
    <password>novell</password>

  </local-tx-datasource>

</datasources>
```

Modify the standardjaws.xml or jaws.xml configuration file

The **standardjaws.xml** file is the standard descriptor for mapping Container Managed Persistence (CMP) entity EJBs. If you plan to use the standard configuration for EJB processing using CMP, you need to edit this file. If you want to use a custom configuration for mapping CMP entity EJBs, you need to edit the **jaws.xml** file instead.

Whether you use standardjaws.xml or jaws.xml, you need to set the <datasource> and <type-mapping> elements, as shown below:

```
<jaws>
  <datasource>java:/MySqlDS</datasource>
  <type-mapping>mySQL</type-mapping>
</jaws>
```

Copy the file to the META-INF directory within the EJB jar file.

Modify the standardjbosscmp-jdbc.xml or jbosscmp-jdbc.xml configuration file

The **standardjbosscmp-jdbc.xml** file is the standard descriptor for configuring the JBoss CMP container. If you want to use a custom configuration for the JBoss CMP container, you can edit the **jbosscmp-jdbc.xml** file instead.

Whether you use standardjbosscmp-jdbc.xml or jbosscmp-jdbc.xml file, you need to set the <datasource> and <datasource-mapping> elements, as shown below:

```
<jbosscmp-jdbc>
  <defaults>
    <datasource>java:/MySqlDS</datasource>
    <datasource-mapping>mySQL</datasource-mapping>
  </defaults>
</jbosscmp-jdbc>
```

Copy the file to the META-INF directory within the EJB jar file.

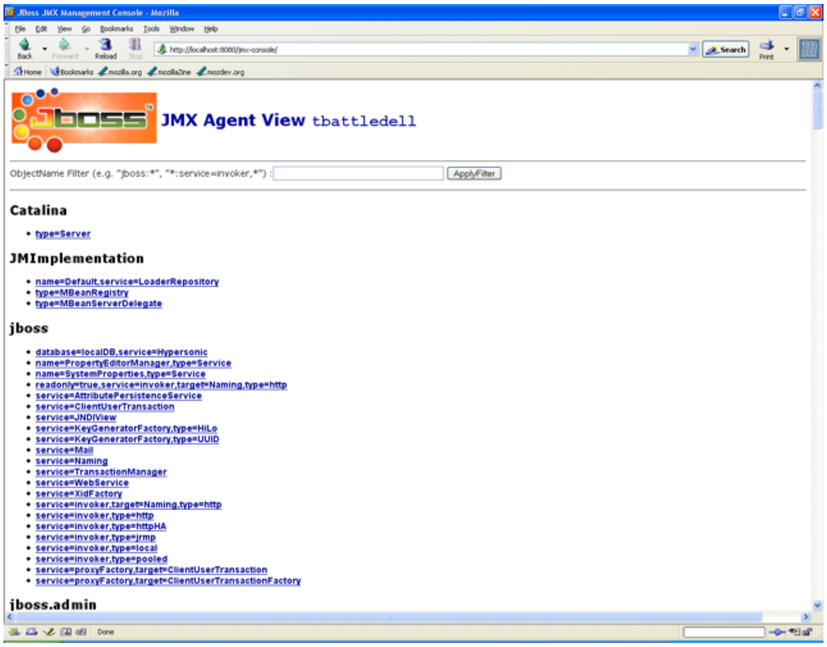
Basic server management

JBoss comes with a JMX Console that provides simple tools for managing the JMX MBeans that make up the running server. The JMX Console lets you display configuration settings, modify these settings, and start or stop components.

To display the JMX Console:

1 Type **http://localhost:8080/jmx-console** in a browser window.

When you type this URL, you should see a window that looks like this:


JBoss JMX Management Console - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8080/jmx-console/
Search
Print
Home Bookmarks mozilla.org mozillaZine mozdev.org
ApplyFilter
Done

JMX Agent View tbattledell

ObjectName Filter (e.g. "jboss:*", "*:service=invoker,*") :

Catalina

- type=Server

JMImplementation

- name=Default,service=LoaderRepository
- type=MBeanRegistry
- type=MBeanServerDelegate

jboss

- database=localDB,service=Hypersonic
- name=PropertyEditorManager,type=Service
- name=SystemProperties,type=Service
- readonly=true,service=invoker,target=Naming,type=http
- service=AttributePersistenceService
- service=ClientUserTransaction
- service=JNDIView
- service=KeyGeneratorFactory,type=HiLo
- service=KeyGeneratorFactory,type=UUID
- service=Mail
- service=Naming
- service=TransactionManager
- service=WebService
- service=XidFactory
- service=invoker,target=Naming,type=http
- service=invoker,type=http
- service=invoker,type=httpHA
- service=invoker,type=jrmp
- service=invoker,type=local
- service=invoker,type=pooled
- service=proxyFactory,target=ClientUserTransaction
- service=proxyFactory,target=ClientUserTransactionFactory

jboss.admin

NOTE: JBoss also comes with a Web console that uses a Java applet to handle presentation. To start this facility, type **http://localhost:8080/web-console/** in a browser window.

Tips for exteNd users

This section provides some useful tips on server management for exteNd users. It lists server management techniques exteNd users would perform in the SMC and shows how one might perform these techniques in the JMX console:

exteNd Application Server technique	**JMX console equivalent**
Displaying basic server configuration information (SMC Configuration section)	**jboss.system** section ◆ **type=server** gives build and server startup information and also allows you to shutdown the server. ◆ **type=ServerConfig** provides information about the current JBoss configuration.
Managing deployed objects (SMC Deployment section)	**jboss.web.deployment** section lists all deployed WAR files.
Displaying the JNDI naming tree (SMC Deployment Section)	**jboss** section ◆ **service=JNDIView** lets you see a view of the current names in the JNDI tree. NOTE: To see this view, you need to scroll down to the **list** function near the bottom of the page, and click **invoke**.
Managing resource adaptors (SMC Deployment Section)	**jboss.jca** section provides information about resource adaptors.

exteNd Application Server technique	JMX console equivalent
Statistics (SMC Monitor Section)	**jboss** section ◆ **service=TransactionManager** lists some transaction statistics. **jboss.jca** section ◆ **name=*datasource*,service=ManagedConnectionPool** lists some transaction statistics. **jboss.management.local** section ◆ **J2EEServer=Local,j2eeType=JTAResource,name=TransactionManager** lists some transaction statistics. **jboss.system** section ◆ **type=serverinfo** gives JBoss thread information. **jboss.web** section ◆ **Host=localhost,path=/jmx-console,type=Manager** shows the number of JMX-console active sessions and run statistics. ◆ **name=http-0.0.0.0-8080,type=threadpool** provides a list of Tomcat threads. ◆ **name=http-0.0.0.0-8080,type=GlobalRequestProcessor** provides some request statistics.

Where to find more information

The following table provides links to useful topics on installation, configuration, and server management in the JBoss documentation:

Topic	Where to find more information
JBoss installation	Installing and Building the JBoss Server
Basic JBoss configuration	The JBoss 4 Server - A Quick Tour
Database configuration	Using other Databases
JBoss architecture	The JBoss JMX Microkernel
JNDI	Naming on JBoss
Transactions	Transactions on JBoss
Security	Security Security on JBoss
Web Services	J2EE Web Services

3 Deploying an exteNd Director Project

This chapter provides information about creating, modifying, and deploying an exteNd Director project to a JBoss Application Server. It includes these sections:

- About the JBoss Deployment environment
- About the deployment steps
- Setting up the JBoss environment for an exteNd deployment
- Setting up an exteNd Director project for a JBoss deployment
- Deploying to JBoss

About the JBoss Deployment environment

Use the following table as a quick-reference to help you find the JBoss equivalent for the exteNd Application Server features you're accustomed to working with.

Topic	On the Novell exteNd Application Server use	On the JBoss Application Server use
Deployment tools	◆ Director Designer ◆ SilverCmd	◆ **Director Designer**—Can be used for local deployments only. A local deployment is one where the JBoss server and Director Designer are on the same machine. ◆ **Native JBoss**—Can be used for local and remote deployments. Copy the deployment archive (EAR, WAR, etc.) to the server's **deploy** directory. For example, to deploy to the default server, you'd copy the archive to: `server\default\deploy` 📖 For more information, see "Deploying to JBoss" on page 10.

Topic	On the Novell exteNd Application Server use	On the JBoss Application Server use
Deployment Plan/ Descriptors	◆ Requires a deployment plan. ◆ Generated automatically by the Director Designer and editable via the Deployment Plan Editor.	◆ Requires application deployment descriptors. They are include: EAR—META-INF\jboss-app.xml WAR—WEB-INF\jboss-web.xml SAR—jboss-service.xml EJB—jboss.xml ◆ DTDs are located in the JBoss \docs\dtd directory. Director Designer generates the file that is appropriate for the archive type whenever you create a new project that specifies one of these realms: **JBoss** and **JBossLDAP**. 📖 For more information, see "What happens when you use a JBoss realm" on page 8. If you modify an existing project, you'll need to manually create these files. See "About the JBoss deployment descriptor entries" on page 9 for information about the required entries.
Hot deploy	Supported for all deployments. Server does not require a shutdown.	Supported for all deployments. Server does not require a shutdown.
Quick deploy	Supported by the Director Designer deployment tool. Quick deploys when specified in the project's deployment settings.	Called Exploded Deployment. ◆ **Director Designer**—Will deploy an exploded archive to the specified directory when the Quick deploy check box (under Deployment Settings) is checked. ◆ **Native JBoss**—Copy the exploded archive to the server's **deploy** directory.
Undeploy	◆ Director Designer ◆ SilverCmd ◆ SMC	◆ **Director Designer**—Use to undeploy locally. ◆ **Native JBoss**—Use to undeploy locally or remotely. Just delete the archive from the server's **deploy** directory.

About the deployment steps

Use the table below as a checklist for deploying an exteNd Director project to the JBoss Application Server environment:

Step	What to do	Notes
1	Configure the JBoss environment	📖 For more information, see "Setting up the JBoss environment for an exteNd deployment" on page 3.
2	**Create** or **modify** the exteNd Director project to deploy	📖 For more information, see "Setting up an exteNd Director project for a JBoss deployment" on page 4.

Step	What to do	Notes
3	Deploy	For more information, see “Deploying to JBoss” on page 10. If you are migrating the project from another server, import any portal data you exported. 📖 See the chapter on moving portal data in the *Portal Guide*.

📖 For more information about moving from a test to production environment, see *Developing exteNd Director Applications*.

Setting up the JBoss environment for an exteNd deployment

Before deploying an exteNd Director project to JBoss, make sure that you:

Step	What to do	Notes
1	Create a JBoss datasource file for your exteNd Director database.	📖 For more information, see “Working with databases” on page 18.

2	Verify that the datasource’s JDBC driver is in the JBoss server’s **lib** directory.	On **Linux**, you’d copy the driver from this exteNd directory:
		`opt/novell/extend5/mysql/jdbc`
		to a JBoss directory like this:
		`/server/default/lib`

		On **Windows**, you’d copy the driver from this exteNd directory:
		`\MySQL\jdbc`
		to a JBoss directory like this:
		`\server\default\lib`

3	Make sure that the exteNd copies of these files are available to the Java runtime:
	◆ Phaos_Crypto_FIPS.jar
	◆ Phaos_Security_Engine.jar
	◆ Phaos_SSLava.jar
		On **Linux**, copy the Phaos files from this exteNd directory:
		`Common/jre/lib/ext`
		to the JRE directory:
		`JDK/jre/lib/ext`

		On **Windows**, copy the files from this exteNd directory:
		`Common\jre\lib\ext`
		to the JRE directory:
		`JDK\jre\lib\ext`

	NOTE: If you are developing on a Windows platform and deploying to a JBoss server running on UNIX, copy the Phaos_Crypto_FIPS_UNIX.jar from the Common\lib\other directory and rename it Phaos_Crypto_FIPS.jar.	

Step	What to do	Notes

4	**For the all configuration only**
	Add an entry for the JBoss Castor.jar to the server's run.sh or run.bat file.
	You'll find the JBoss Castor.jar in jboss/server/all/deploy/snmp-adaptor.sar.
	NOTE: If you don't add this you will encounter ClassCastExceptions because of conflicts between multiple versions of the Castor.jar.
		On **Linux**, add the following right after the line:
		`"JBOSS_ENDORSED_DIRS="$JBOSS_HOME/lib/endorsed"`
		`#####################################`
		`# Novell extend settings for all config #`
		`###########################################`
		`if [ "$*" = "-c all" ]; then`
		`if [ "x$JBOSS_CLASSPATH" = "x" ]; then`
		`JBOSS_CLASSPATH=$JBOSS_HOME/server/all/deploy/snmp-adaptor.sar/castor.jar`
		`else`
		`JBOSS_CLASSPATH=$JBOSS_CLASSPATH:$JBOSS_HOME/server/all/deploy/snmp-adaptor.sar/castor.jar`
		`fi`
		`fi`

		On **Windows**, add the following right after the line:
		`"JBOSS_ENDORSED_DIRS="$JBOSS_HOME/lib/endorsed"`
		`rem########################################`
		`rem#Novell extend settings for all config #`
		`rem#########################################`
		`if "%*" == "-c all" (`
		`if "%JBOSS_CLASSPATH%" == "" (`
		`set JBOSS_CLASSPATH=%JBOSS_HOME%\server\all\deploy\snmp-adaptor.sar\castor.jar`
		`) else (`
		`set JBOSS_CLASSPATH=%JBOSS_CLASSPATH%;%JBOSS_HOME%\server\all\deploy\snmp-adaptor.sar\castor.jar`
		`)`
		`)`

5	(Optional) To enable contextual searching, configure Autonomy.	See the section on configuring your environment for conceptual searching in the *Content Search Guide*.

Setting up an exteNd Director project for a JBoss deployment

The procedures that you'll follow for setting up your project for JBoss deployment vary depending on the type of project you are working with.

- Working with existing projects
- Working with new projects

Working with existing projects

If you are migrating an existing project from a different application server type to JBoss, you'll:

Step	Action	Notes
1	**Backup** your existing project	Export the portal data (such as container, shared, personal pages, and portlets). 📖 See the chapter on moving portal data in the *Portal Guide*.

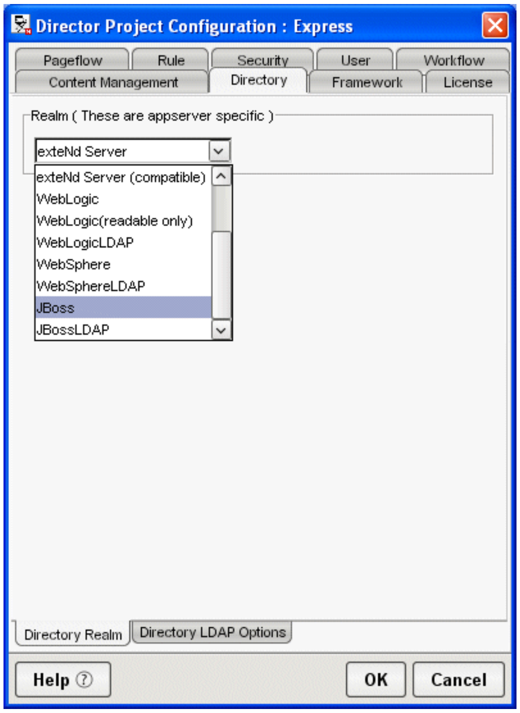
Step	Action	Notes
2	Change the project's **security realm**	Follow the procedure outlined in "To change the project's security realm:" on page 5.
3	Change the project's **Framework Datasource specification**	Follow the procedure outlined in "To change the project's Director Framework Datasource for JBoss:" on page 6.
4	**Create** a jboss-app.xml file for an EAR project or a jboss-web.xml file for a WAR project	The exteNd deployments require only the <loader-repository> element described in "About the JBoss deployment descriptor entries" on page 9. For more information about the contents of the JBoss deployment descriptors in general, see your JBoss documentation.
5	Rebuild and archive the project	See the chapter on projects and archives in *Utility Tools*.

NOTE: After you deploy the project on the JBoss application server, you'll need to import the portal pages and portlets that you exported in Step 1 above. See the chapter on moving portal data in the *Portal Guide* for instructions.

Procedures for changing existing projects

➢ To change the project's security realm:

1. With the project open, choose **Project>Director>Configuration**.
2. Choose **Directory**.



3 Choose **JBoss** or **JBossLDAP** from the dropdown.

Realm configuration	Description
JBoss	Read and write access to the JBoss application server security realm. This realm uses the exteNd Director database as the users store.
JBossLDAP	Read and write access to the JBoss application server realm using an LDAP server. This realm authenticates the user through a JBoss JAAS module. If you choose this realm, click the Directory LDAP Options tab and complete that panel as well. For more information, see *Developing exteNd Director Applications*.

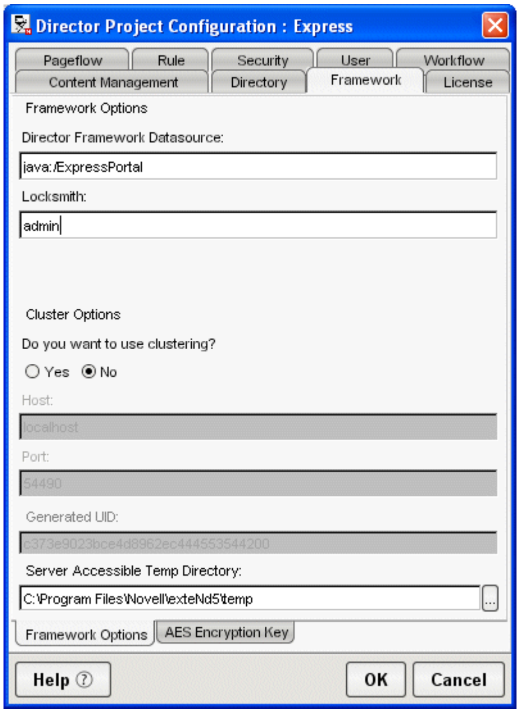
4 Click **OK**.

NOTE: When you choose one of the JBoss realms, the Director Designer modifies the JBoss server's /conf/login-config.xml file. You can learn more about what it adds in the "What happens when you use a JBoss realm" on page 8.

➢ To change the project's Director Framework Datasource for JBoss:

1 With the project open, choose **Project>Director>Configuration**.

2 Choose **Framework**.



Framework Options

Director Framework Datasource: java:/ExpressPortal

Locksmith: admin

Cluster Options

Do you want to use clustering? ○ Yes ● No

Host: localhost

Port: 54490

Generated UID: c373e9023bce4d8962ec444553544200

Server Accessible Temp Directory: C:\Program Files\Novell\exteNd5\temp

3 Make sure the Director Framework Datasource is prefaced with the string **java:/**. For example:

```
java:/ExpressPortal
```

Working with new projects

There are two exteNd Director project specifications that require JBoss Application Server-specific entries. They occur when:

- Choosing a security realm.

The security realm choices panel is described in “To specify the new project’s security realm:” on page 7.

- Specifying the Director Framework datasource.

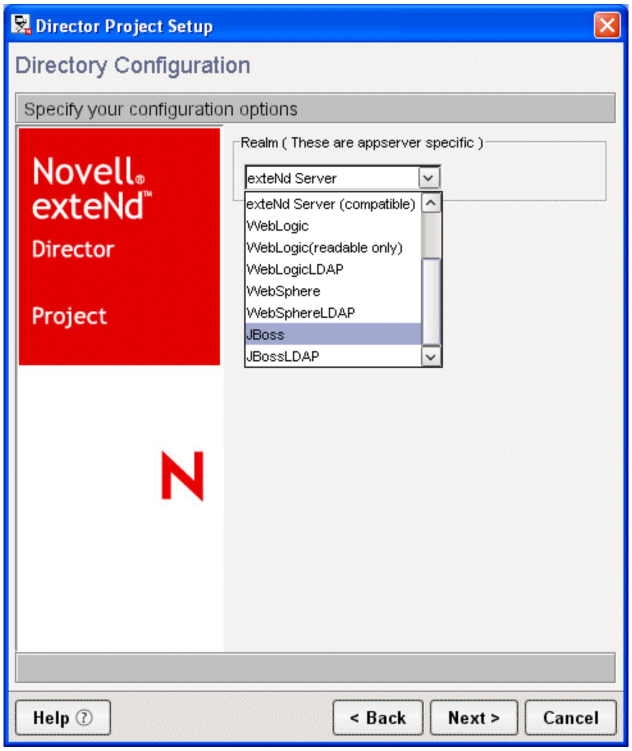
  The Director Framework datasource panel is described in “To specify the new project’s Director Framework Datasource:” on page 7.

NOTE: The instructions in this section describe just the panels where you’ll make JBoss-specific choices. See *Developing exteNd Director Applications* for information about other project configuration options.

JBoss-specific new project specifications

➢ To specify the new project’s security realm:

- When prompted for the project’s security realm



Realm (These are appserver specific): exteNd Server

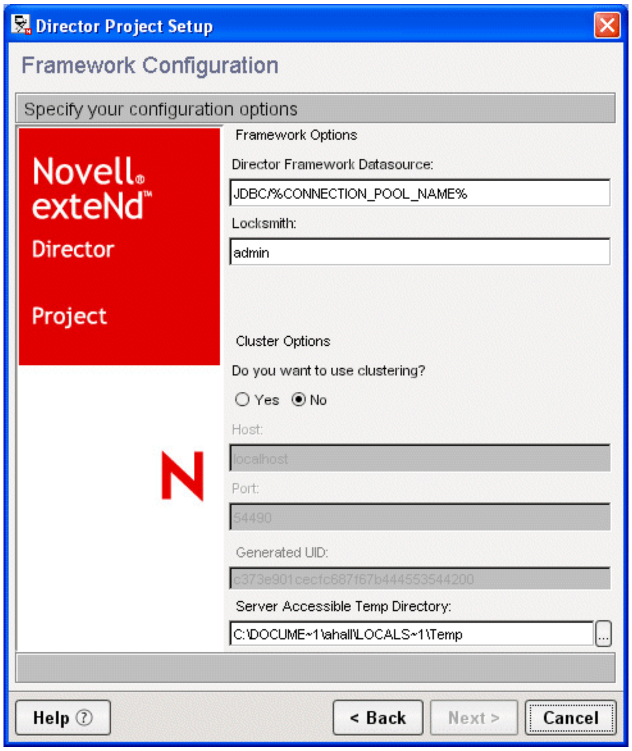
choose one of the following:

Realm configuration	Description
JBoss	Read and write access to the JBoss application server security realm. This realm uses the exteNd Director database as the users store.
JBossLDAP	Read and write access to the JBoss application server realm using an LDAP server. This realm authenticates the user through a JBoss JAAS module.

📖 For more information about the JBoss realms, see “What happens when you use a JBoss realm” on page 8.

➢ To specify the new project’s Director Framework Datasource:

- When prompted for the new project’s Director Framework Datasource:



Framework Configuration

Specify your configuration options

Framework Options

Director Framework Datasource: JDBC/%CONNECTION_POOL_NAME%

Locksmith: admin

Cluster Options

Do you want to use clustering?

○ Yes ● No

Host: localhost

Port: 54490

Generated UID: c373e901cecfc687f67b444553544200

Server Accessible Temp Directory: C:\DOCUME~1\ahall\LOCALS~1\Temp

Complete the Director Framework Datasource as follows:

- Replace the String **JDBC/** with **java:/**

  This is required because JBoss puts all deployed datasources in the java domain of the JNDI tree.

- Replace **%CONNECTION_POOL_NAME%** with the name of the exteNd Director databases. For example:

  ```
  java:/ExpressPortal
  ```

What happens when you use a JBoss realm

When you choose one of the JBoss realms, the Director Designer:

- **Creates** the JBoss-specific deployment descriptor (for new projects only)

  📖 For more information, see About the JBoss deployment descriptor entries next.

- **Adds** an <application-policy> node to the server’s conf\login-config.xml file

  If your JBoss deployment is a **remote** deployment, then the attempt to modify the **login-config.xml** fails and you’ll need to add the <application-policy> node manually.

  📖 For more information on how the Director Designer generates the <application-policy> node for local deployments, see About the JBoss deployment descriptor entries next.

About the JBoss deployment descriptor entries

When the JBoss deployment descriptors are created, entries are added for the <loader-repository> node. They are described in the following table:

Archive type	Deployment descriptor entry
WAR	In WEB-INF/jboss-web.xml: <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE jboss-web PUBLIC "-//JBoss//DTD Web Application 2.4//EN" "http://www.jboss.org/j2ee/dtd/jboss-web_4_0.dtd"> <jboss-web> <class-loading java2ClassLoadingCompliance="true"> <loader-repository>com.novell:loader=*MyWar.war* <loader-repository-config>java2ParentDelegation=true</loader-repository-config> </loader-repository> </class-loading> </jboss-web>
EAR	In META-INF/jboss-app.xml: <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE jboss-app PUBLIC "-//JBoss//DTD J2EE Application 1.4//EN" "http://www.jboss.org/j2ee/dtd/jboss-app_4_0.dtd"> <jboss-app> <loader-repository>com.novell:loader=*MyEAR.ear* <loader-repository-config>java2ParentDelegation=true</loader-repository-config> </loader-repository> </jboss-app>

About the <application-policy> node

The JBoss Application Server needs to know which JAAS module to use for logging an exteNd user in to JBoss. It uses the <application-policy> entry in the **/conf/login-config.xml** file to determine this. The Director Designer adds an <application-policy> node to the server's login-conf.xml file, at deployment time, when your project includes both of the following:

- A JBoss-specific realm
- A JBoss server profile

NOTE: When the JBoss server and Director Designer are not on the same machine, the attempt to add the <application-policy> node fails and you must add it manually.

The Director Designer generates the <application-policy> node (per realm) as follows:.

Realm	Default <application-policy> node
JBoss realm	<application-policy name="extend"> <authentication> <login-module code="com.sssw.fw.server.jboss.jaas.LoginModule" flag="required" /> </authentication> </application-policy>
JBossLDAP	<application-policy name="extendLDAP"> <authentication> <login-module code="com.sssw.fw.server.jboss.jaas.JndiLoginModule" flag="required" /> </authentication> </application-policy>

The value that Director Designer uses for the <application-policy>'s name attribute is taken from the **DirectoryService/jboss/application-policy** property (located in the project's DirectoryService-conf\config.xml file). The value of this property is automatically set to **extend** or **extendLDAP** when a new project is created in one of the JBoss realms, or when the realm of an existing project is changed to one of the JBoss realms.

Deploying to JBoss

After you've set up your JBoss environment and the exteNd Director project, you are ready to deploy.

NOTE: To use the Director Designer to deploy to a JBoss server, the Director Designer must be installed on the same machine as the JBoss server.

This section describes the steps for deploying to both a local and remote JBoss application server.

Step	What to do	Notes
1	Build and archive the project	See the chapter on projects and archives in *Utility Tools*.
2	**For a local deploy only** Create a JBoss server profile and deployment settings for your project	See the procedure "To create JBoss deployment settings (for local deployments only):" on page 11.
	For a remote deploy only Add an <application-policy> node to the server's login-config.xml for each project that you deploy	See About the <application-policy> node. The <application-policy> node you add should follow the same rules as Director Designer uses.
3	Deploy the archive	For local deployments, see the chapter on deploying a project in *Utility Tools*. For remote deployments, copy the archive to the server's \deploy directory.
4	Test the deployment	Open a browser. Type the URL for the exteNd Director application context. For example: `http://localhost:8080/ExpressPortal`

Redeploying When you redeploy a project to the JBoss all or default configurations, you might encounter a java.io.NotSerializableException. This message is informational only. If you want to prevent this error from occurring you can do the following before redeploying:

1. Remove the server's tmp and work directories.
2. Restart the server.

➢ To create JBoss deployment settings (for local deployments only):

1 In the Director Designer, choose **Project>Deployment Settings**.

Option	What to do
Profile name	Select a JBoss server profile from the list or click **New** to create a new profile. If you create a new profile, complete the fields as follows: ◆ **Server profile**—Provide a meaningful profile name. ◆ **Server type**—Select the JBoss entry. ◆ **Deployment tools directory**—Browse to the \deploy directory for the server configuration that you want to deploy to. ◆ **Rapid deployment directory**—Browse to the \deploy directory for the server configuration that you want to deploy to. ◆ **Server name**—Specify the host name:port or the IP address. 📖 For more information on server profiles, see the chapter on deploying a project in *Utility Tools*.
Use this server profile as the default for all projects	Select this option to make the current server profile the default profile for your projects.
User name and Password	If you have a secure server, fill in the **User name** and **Password** text boxes with an authorized user name and password for the server.

2 To enable rapid deployment, select the **Deployment Info** tab.

Option	What to do
Enable Rapid Deployment	Check this box when you want to deploy an exploded archive in the server's \deploy directory. NOTE: This feature is recommended for testing.

3 Click **OK** to store the deployment settings with the project file.

4 Deploying an exteNd Composer Project

This chapter provides information about creating, modifying, and deploying an exteNd Composer project to a JBoss Application Server. It includes these sections:

- Composer and the JBoss Runtime Environment
- About Composer Deployment Options
- Connection Pooling for Process Manager

Composer and the JBoss Runtime Environment

Deployment and execution of Composer-built services on the JBoss application server relies on the existence and proper placement of certain files in the JBoss directory structure. This section explains the runtime significance of these files, where the files are located, and why you might want to hand-edit them under certain circumstances.

Key Runtime Files

Composer services rely, for their execution and lifecycle management, on an execution engine called the Composer Enterprise Server. This engine is contained in an archive called exteNdComposer.ear.

At installation time, exteNdComposer.ear is placed under \server\default\deploy as well as \server\all\deploy. (Any other copies of this file that you may see on disk are leftover temporary copies and may be deleted.) When you run your JBoss server in the default configuration, the copy of exteNdComposer.ear under \server\default\deploy is active. When you run your JBoss server in the "all" configuration, the copy of exteNdComposer.ear under \server\all\deploy is active.

In order for the Composer Enterprise Server runtime to function correctly in the JBoss environment, three additional files must be present.

- xconfig.xml
- users.properties
- roles.properties

The locations and functions of these files are shown in the following table.

File	Location relative to %JBOSS_HOME%	Purpose
exteNdComposer.ear	\server\all\deploy and \server\default\deploy	This is Composer's server-side execution engine (otherwise known as Composer Enterprise Server)

File	Location relative to %JBOSS_HOME%	Purpose
xconfig.xml	\server\all\conf and \server\default\conf	Runtime configuration file for Composer Enterprise Server. Contains host-name and port information. NOTE: A file by the same name is included under \Composer\Designer\bin in the design-time installation. The design-time file is not identical to the runtime version.
roles.properties	\server\all\conf and \server\default\conf	Associates usernames with roles. Composer uses this file to find the user associated with “ComposerAdministrators.”
users.properties	\server\all\conf and \server\default\conf	Stores username/password associations. Composer uses this file to establish credentials for deploying to the server.

Additional comments about these files follow.

xconfig.xml

This configuration file is consulted by Composer Enterprise Server upon startup and shutdown (and is overwritten on shutdown; therefore, do not make hand-edits to the file while Composer Enterprise Server is running). Several JBoss-specific pieces of information are located in this file, as indicated below. (Note that the locations of these items are specified in *XPath notation* in the lefthand column of the table.)

Element (in XPath Notation)	Contents
JBOSS_40/ADMIN_HOST	Stores the host name (such as “localhost”) you provided at install time. If you later rename the host, you must edit this element accordingly.
JBOSS_40/ADMIN_PORT	This is the port information you supplied at install time (for example, “8080”).
JBOSS_40/JBOSS_HOME	The JBoss installation root folder (such as “D:\jboss-4.0.1”).
JBOSS_40/STAGING_LOCATION	Path to the staging folder where Composer will write archives and pre-deployment files.
JBOSS_40/DEPLOY_INPUT	Location, within the exteNdComposer.ear archive, of the DeployInput.jsp page (which is the browser page that prompts you for credentials upon deployment of a web app).
JBOSS_40/DEPLOY_HANDLER	Name of the helper class that carries out final steps of deployment. You should not edit this element.

A typical JBOSS_40 entry in xconfig.xml looks like this:

```
<JBOSS_40>
      <ADMIN_HOST>localhost</ADMIN_HOST>
      <ADMIN_PORT>8080</ADMIN_PORT>
      <JBOSS_HOME>D:\jboss-4.0.1</JBOSS_HOME>
```

```
            <STAGING_LOCATION>D:\jboss-4.0.1\server\all\tmp\deploy</STAGING_LOCATION>
         <DEPLOY_INPUT>deploy/jb/DeployInput.jsp</DEPLOY_INPUT>
         <DEPLOY_HANDLER>Composer.jb.DeployHandler</DEPLOY_HANDLER>
   </JBOSS_40>
```

If you alter your server's host name or default port, or rename the JBoss installation root directory, you should make appropriate edits of the associated elements (see above) in xconfig.xml using a text editor or XML editor. Normally, these are the only kinds of manual edits you will ever need to make.

roles.properties and users.properties

For its default file-based access-control functionality, JBoss uses a simple properties-file-based login module to read information from simple text files. For Composer deployments, the files of interest are "users.properties,"which contains username-to-password mappings, and "roles.properties," which contains username-to-role mappings.

NOTE: The names of these properties files may be overridden by the `usersProperties` and `rolesProperties` options in the JBoss login-conf.xml file. Consult JBoss documentation for detailed information on security-configuration options.

The users.properties file uses a format:

```
username1=password1
username2=password2
...
```

to define all valid usernames and their corresponding passwords.

The roles.properties file uses a format:

```
username1=role1,role2, ...
username1.RoleGroup1=role3,role4, ...
username2=role1,role3, ...
```

to define the sets of roles for valid usernames.

If your server\default\conf or server\all\conf directory (as appropriate) does *not* contain these files, create them in a text editor and include the following content:

- For **users.properties**, the content should be `novell=novell` (assuming you accepted the username and password defaults when you installed Composer Enterprise Server).
- For **roles.properties**, the content should be `novell=ComposerAdministrators` (assuming the username, as given in **users.properties**, is "novell").

About Composer Deployment Options

No matter which vendor's app server you are using, the exteNd toolset offers two basic ways to deploy web applications built with exteNd Composer:

- Users of exteNd Composer Professional Edition can perform deployments using the Director Designer's utility tools.
- Users of Enterprise Edition may deploy either from Director Designer *or* directly from Composer Designer.

Each of these will now be discussed briefly. For more comprehensive information, consult the deployment chapter of the Composer User's Guide.

Deploying from Professional Edition

Detailed procedure(s) for deploying a Composer project from Director Designer are covered in the deployment chapter of the Composer User's Guide and won't be repeated here. The basic procedure is to create or open an EAR project in Director Designer, import your Composer project into the EAR project, then use Director Designer's wizards to create new servlets, EJBs, JSPs, etc. for use as service triggers, to trigger your Composer services. From that point on, deployment of the EAR proceeds according to the usual exteNd utility tools deployment procedures. (Consult the preceding chapter for additional information on those procedures.)

Deploying from Enterprise Edition

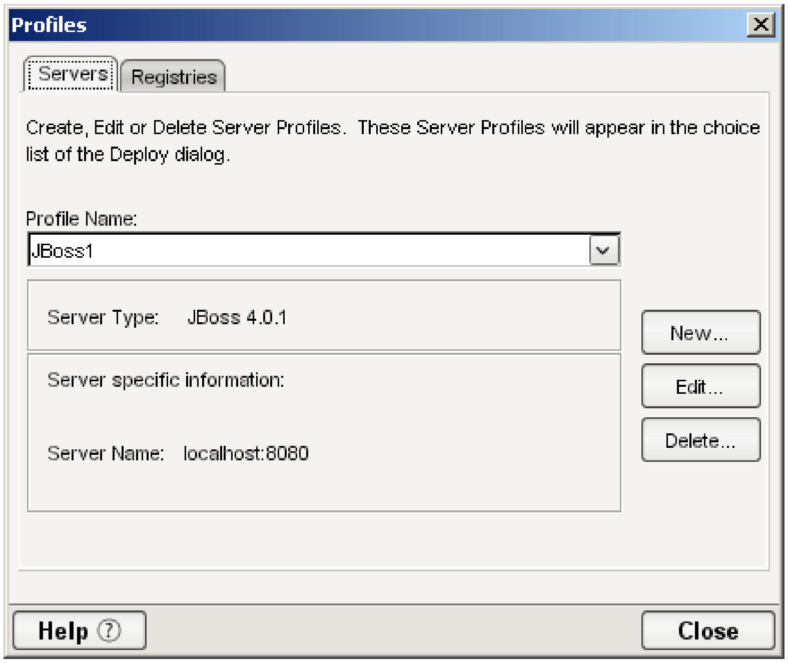
Enterprise Edition users of Composer have the option of creating Deployment xObjects and deploying projects to JBoss directly from the Composer Designer work environment. Deployment xObjects contain the information Composer Designer needs in order to build service triggers for you and package them appropriately in your project EAR at deploy time. Composer Enterprise Edition offers an easy GUI for configuring service triggers of various kinds, associating triggers in one-to-one or many-to-one fashion with the services inside a project, assigning target URIs, specifying "run in role" information, specifying non-code resources (such as image files) to be added to the deployment archive, specifying external JARs for copackaging into the deployment archive, etc. All of this information is stored in the Deployment xObject. For detailed information on Deployment xObjects, consult your Composer User's Guide.

When you deploy an exteNd Composer project using the exteNd Composer Designer UI, you'll follow these steps:

Step	What to do	Notes
	Create a server profile for the target JBoss server (if you haven't already done so).	See To create a JBoss server profile in Composer:, below.
1	Create a Deployment xObject for the project.	This is analogous to creating the deployment settings in a Director project.
2	Use the **File > Deploy Project** command to begin deployment.	This brings up a dialog in which you can verify your deployment options. (This is where you can choose to deploy according to one particular xObject out of many that might be stored in the project.) Make any needed adjustments, then click the OK button to begin deployment.
3	Authenticate to the server via a webform that appears automatically in your browser.	Enter the appropriate credentials (per the discussion of users.properties, above). For details (including a screenshot) on this, see To deploy your project to JBoss from the Composer Enterprise Edition design-time environment:, further below.
4	Fill in the "Select a Composer EAR to deploy" webform (which appears automatically).	You'll generally accept the default info. When you click the Submit button, files are copied into the appropriate locations in the JBoss folder hierarchy.
5	Test the deployment.	Open a browser. Type the URL for the application. For example, if your Deployment xObject is named GetInfo and the service trigger is a servlet called Request, your service URL might be: `http://localhost:8080/GetInfo/Request`

➢ To create a JBoss server profile in Composer:

1 In the Composer Designer, choose **Tools** > **Profiles** from the main menubar. A dialog appears.



Create, Edit or Delete Server Profiles. These Server Profiles will appear in the choice list of the Deploy dialog.

Profile Name: JBoss1

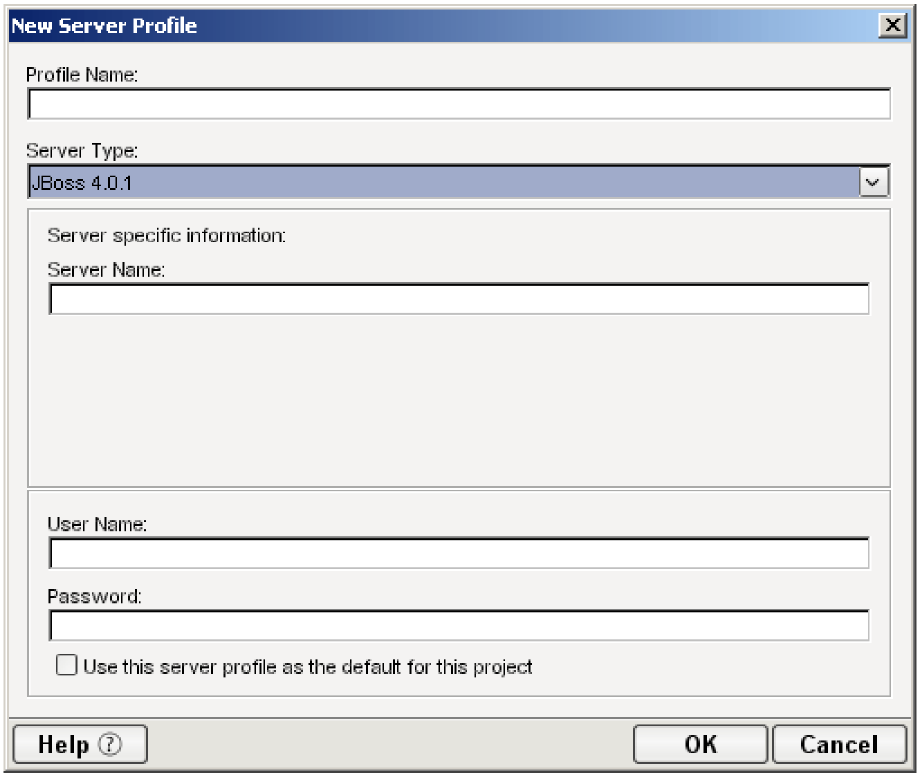
Server Type: JBoss 4.0.1

Server specific information:

Server Name: localhost:8080

2 Bring the **Profiles** tab to the front, if it is not already the foremost tab.

3 In the **Profiles** tab, click the **New...** button. A dialog appears as follows.



Profile Name:

Server Type: JBoss 4.0.1

Server specific information:

Server Name:

User Name:

Password:

□ Use this server profile as the default for this project

4 In the **New Server Profiles** dialog, supply the following info:,

Option	What to do
Profile Name	Provide the (arbitrary) name by which you wish to identify this profile.
Server Type	Select JBoss 4.0.1 from the dropdown list.
Server Name	Enter the host name and (optionally) port number as follows: `localhost:8080`
User Name	Enter the user name (default: “novell”) that corresponds to the user-name info provided in users.properties and roles.properties. (See discussion at roles.properties and users.properties above.)
Password	Enter the password (such as “novell”) that corresponds to the password info in users.properties. (See discussion at roles.properties and users.properties above.)

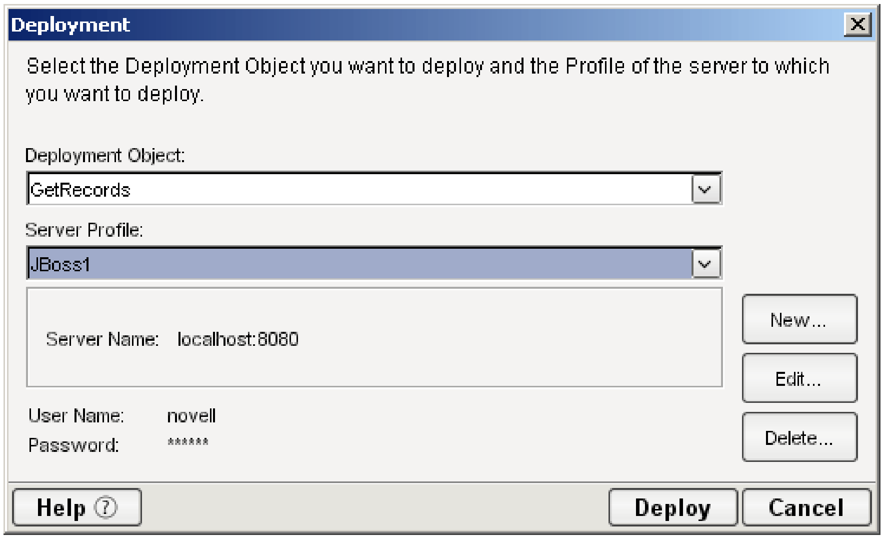
5 Optionally check the “use this server profile as the default” checkbox.

6 Click **OK** to create the profile. Exit out of the **Profiles** dialog (with **Close**) to persist the profile.

➢ **To deploy your project to JBoss from the Composer Enterprise Edition design-time environment:**

NOTE: Be sure your JBoss server is running before attempting to deploy a project from Composer Enterprise Edition. Note that if you start the server in a default configuration, Composer will deploy your project EAR to the server\default\deploy directory. If your JBoss server is running in “all” mode, Composer deploys files to the server\all\deploy directory.

1 Use the **File** > **Deploy Project** command in Composer’s main menubar to begin deployment. A dialog will appear:



Select the Deployment Object you want to deploy and the Profile of the server to which you want to deploy.

Deployment Object: GetRecords

Server Profile: JBoss1

Server Name: localhost:8080

User Name: novell

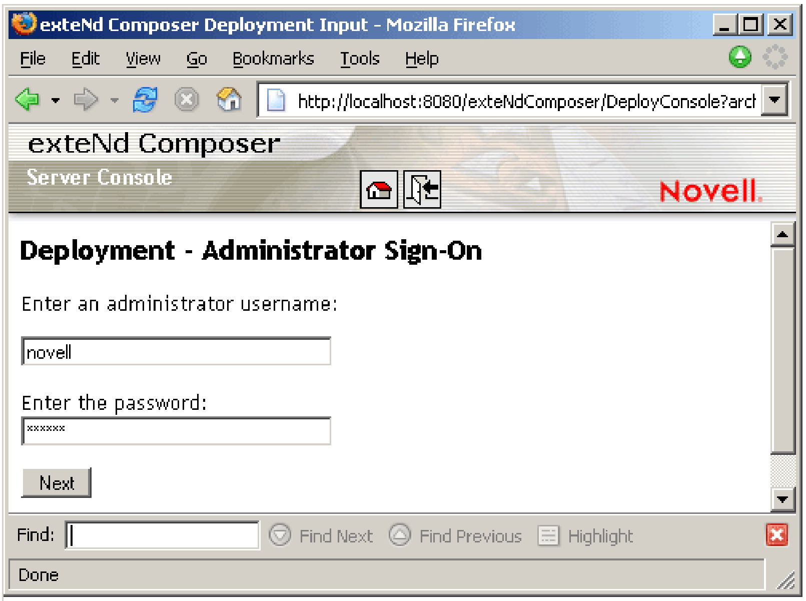
Password: ******

2 Using the dropdown list under **Deployment Object**, choose the Deployment xObject that describes the configuration for this deployment. (If there is just one Deployment xObject in the project, accept the default.)

3 Under **Server Profile**, select the server profile that corresponds to the target server in question.

4 (Optional) If you need to, you can create, edit, or delete a server profile from within this dialog by using the **New**, **Edit**, or **Delete** buttons on the right.

5 Click the **Deploy** button to continue the deployment process. A progress thermometer will appear, followed by the opening of an authenticaton form in your browser:


exteNd Composer Deployment Input - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
exteNd Composer
Server Console
Novell.
Next
Find: Find Next Find Previous Highlight
Done

http://localhost:8080/exteNdComposer/DeployConsole?arch

Deployment - Administrator Sign-On

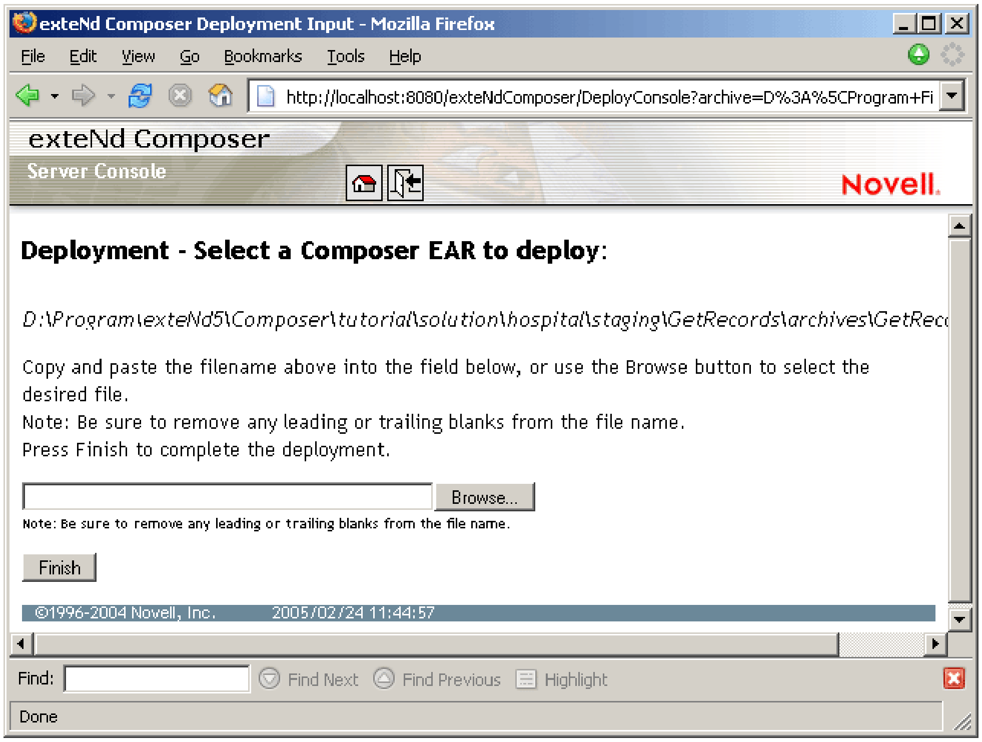
Enter an administrator username:

novell

Enter the password:

******

6 Enter user name and password values appropriate to those stored in roles.properties and users.properties. A new page appears in the browser:


exteNd Composer Deployment Input - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
exteNd Composer
Server Console
Novell.
Browse...
Finish
Find: Find Next Find Previous Highlight
Done

http://localhost:8080/exteNdComposer/DeployConsole?archive=D%3A%5CProgram+Fi

Deployment - Select a Composer EAR to deploy:

D:\Program\exteNd5\Composer\tutorial\solution\hospital\staging\GetRecords\archives\GetRec

Copy and paste the filename above into the field below, or use the Browse button to select the desired file.
Note: Be sure to remove any leading or trailing blanks from the file name.
Press Finish to complete the deployment.

Note: Be sure to remove any leading or trailing blanks from the file name.

©1996-2004 Novell, Inc. 2005/02/24 11:44:57

7 In the text field provided, enter the complete path to the EAR you are deploying. Composer provides the pertinent path in the upper portion of this page, in italics; you can simply copy and paste that text into the text field.

8 Click **Finish**. Files will be copied to the \deploy directory of the currently running instance of JBoss.

Connection Pooling for Process Manager

Composer's Business Process Manager functionality depends on database connection pooling. You will need to accomplish the following setup steps if you intend to use BPM features.

➢ To add a connection pool to JBoss:

1. In the **%JBOSS_HOME%\docs\examples\jca** folder, there is a list of "-ds" files (**oracle-ds.xml**, **mysql-ds.xml,** etc.) for each database that JBoss supports. Copy the appropriate file to the \deploy folder your instance of JBoss will use. Example: If you are using a MySQL database, and you run your server in default mode, copy **mysql-ds.xml** into **%JBOSS_HOME%\server\default\deploy** directory.
2. Edit the **mysql-ds.xml** to contain the correct connection info.
3. Your JBoss server will pick up the pool automatically, but a good practice would be to stop and restart the server at this point.
4. In your browser, navigate to the exteNd Composer Server Console using a URL like:

   `http://localhost:8080/exteNdComposer/Console`
5. On the left side of the main console, click the Process link (under "About Products"). When the content frame refreshes, click the Console button. The Process Manager admin console will pop open in a new window.
6. In the Process Manager console, click the **Configure** button (under "Database Info"). This brings up a screen in which you can choose a database type (e.g., MySQL) and enter a Pool Name, User Name, and Password. Click the **Save** button when you are done.
7. In the Process Manager main console, click the **Start** button to launch the Process Manager engine.

For additional information on working with databases in JBoss, see the section "Working with databases" in Chapter 2 of this guide, as well as the relevant sections of JBoss documentation.

5 Troubleshooting

This chapter provides some troubleshooting tips that you may find helpful when using the JBoss Application Server. Topics include:

- Startup issues
- Deployment issues
- Classloading issues
- Database issues
- Clustering issues

📖 For complete information on configuring and managing the server, see The JBoss 4 Application Server Guide.

Startup issues

Binding to a network address

When running JBoss on Linux under the **all** configuration, which provides **clustering**, you must start JBoss with an **explicit bind** to a network address (using the `-b` parameter with `run.sh`). Make sure you have one of the following for that address:

- An entry of *IP*/*machine-name* in **/etc/hosts**
- An entry in **DNS**

This is particularly important if you run JBoss on a **multi-homed** server (a server that has multiple NICs). Without the `-b` parameter, JBoss will throw exceptions saying that the server could not bind to a given address.

Deployment issues

DeploymentException in exteNd Director Designer

If you are deploying an EAR from the exteNd Director Designer to JBoss, a timing issue between the Designer and JBoss may cause the following DeploymentException to display:

```
No META-INF/application.xml found
```

This exception occurs on the first deployment, but that deployment should succeed anyway. You should not see the exception on subsequent deployments. (PPR 48905)

Classloading issues

About classloading

Unlike the exteNd Application Server, JBoss uses a **flat classloading model**. All classes are loaded by the same classloader.

However, if applications can't share classes because of version conflicts, you need to **scope the classes** from other deployments. Common indications of this are ClassCastException, IllegalAccessErrors, VerifyErrors and, in general, strange behavior that changes as new deployments are added or removed.

The **two levels of scoping** are:

- Isolation from other deployments (which corresponds to how the exteNd Application Server works)
- Isolation with override of server classes

WAR classloading is delegated to the web container (by default, Tomcat) and the web container's classloading rules apply (see the Tomcat documentation for details). But JBoss passes its classloader to the web server as context classloader, and the web container may use this context classloader.

JBoss also uses a trick to avoid classloader locks on server startup.

Scoping classloaders

For **jboss-app.xml** (JBoss EAR descriptor) and **jboss.xml** (JBoss EJB descriptor), add the following descriptor fragment constructs to enable scoped classloading with the deployment classes isolated from other deployments:

```
<loader-repository>
   unique-scope-name:loader=unique-archive-name
</loader-repository>
```

For **jboss-web.xml** (JBoss WAR descriptor), the following template applies:

```
<class-loading>
   <loader-repository>
      unique-scope-name:loader=unique-archive-name
   </loader-repository>
</class-loading>
```

📖 For more information, see the JBoss Wiki page on classloading configuration.

Classloader debugging

Classloader **logging** is no different than any other JBoss logging, which is based on **log4j**. To enable logging for classloaders, edit **log4j.xml** (in server/*xxx*/conf) and add:

```
<appender name="UCL" class="org.apache.log4j.FileAppender">
   <param name="File" value="${jboss.server.home.dir}/log/ucl.log"/>
   <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="[%r,%c{1},%t] %m%n"/>
   </layout>
</appender>

<category name="org.jboss.mx.loading" additivity="false">
   <priority value="TRACE" class="org.jboss.logging.XLevel"/>
   <appender-ref ref="UCL"/>
</category>
```

Note that element `log4j:configuration` must match:

```
(renderer*,appender*,(category|logger)*,root?,categoryFactory?)
```

The order of tags is important.

Database issues

Database connection recovery

If an action on a connection in the pool fails, the server will check all connections in the pool and recreate them if necessary.

JBoss can **test a connection** before handing it out by calling a SQL statement. Add the following element to the **-ds.xml** file:

```
<check-valid-connection-sql>some cheap sql statement</check-valid-connection-sql>
```

Datasource statistics

In the JBoss **JMX Console**, you can view statistics on a specific datasource (such as MyDatasource) by going to the **jboss.jca** section and clicking:

```
name=MyDatasource,service=ManagedConnectionPool
```

Clustering issues

Editing the JGroups configuration

When running JBoss on Windows with clustering (under the **all** configuration), you must edit the **JGroups configuration** for the following files in the **deploy** directory:

- cluster-service.xml
- tc5-cluster-service.xml

Look for this comment and follow the instructions:

```
<!-- UDP: On Windows machines, because of the media sense feature
     being broken with multicast (even after disabling media sense)
     set the loopback attribute to true -->
```