BEA WebLogic Server™

WebLogic Server Command Reference

Release 8.1
Document Revised: December 5, 2002

System Administration Reference

Part Number	Document Revised	Software Version
N/A	December 5, 2002	BEA WebLogic Server Version 8.1

Contents

About This Document

1. weblogic.Admin Command-Line Reference

2. Using the WebLogic Server Java Utilities

3. weblogic.Server Command-Line Reference

About This Document

This document [[introduces BEA WebLogic Server™ features and describes the architecture of applications that run on the WebLogic Server platform.]]

The document is organized as follows:

- Chapter 1, "weblogic.Admin Command-Line Reference,"describes using the `weblogic.Admin` command to configure a WebLogic Server domain from a command shell or a script.
- Chapter 2, "Using the WebLogic Server Java Utilities," describes various Java utilities you can use to manage and troubleshoot a WebLogic Server domain.
- Chapter 3, "weblogic.Server Command-Line Reference," describes how to start WebLogic Server instances from a command shell or from a script.

Audience

This document is written for system administrators and application developers deploying e-commerce applications using the Java 2 Platform, Enterprise Edition (J2EE) from Sun Microsystems. It is assumed that readers are familiar with Web technologies and the operating system and platform where WebLogic Server is installed.

e-docs Web Site

BEA product documentation is available on the BEA corporate Web site. From the BEA Home page, click on Product Documentation.

How to Print the Document

You can print a copy of this document from a Web browser, one main topic at a time, by using the File—Print option on your Web browser.

A PDF version of this document is available on the WebLogic Server documentation Home page on the e-docs Web site (and also on the documentation CD). You can open the PDF in Adobe Acrobat Reader and print the entire document (or a portion of it) in book format. To access the PDFs, open the WebLogic Server documentation Home page, click Download Documentation, and select the document you want to print.

Adobe Acrobat Reader is available at no charge from the Adobe Web site at http://www.adobe.com.

Related Information

- Creating and configuring WebLogic Servers and Domains
- Managing a WebLogic Server Domain
- Administration Console Online Help

Contact Us!

Your feedback on BEA documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the documentation.

In your e-mail message, please indicate the software name and version you are using, as well as the title and document date of your documentation. If you have any questions about this version of BEA WebLogic Server, or if you have problems installing and running BEA WebLogic Server, contact BEA Customer Support through BEA WebSupport at http://www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Usage
Ctrl+Tab	Keys you press simultaneously.
italics	Emphasis and book titles.

Convention	Usage

`monospace text`	Code samples, commands and their options, Java classes, data types, directories, and file names and their extensions. Monospace text also indicates text that the user is told to enter from the keyboard.
	Examples:
	`import java.util.Enumeration;`
	`chmod u+w *`
	`config/examples/applications`
	`.java`
	`config.xml`
	`float`

`monospace italic text`	Placeholders. *Example*: `String CustomerName;`

`UPPERCASE MONOSPACE TEXT`	Device names, environment variables, and logical operators.
	Examples:
	`LPT1`
	`BEA_HOME`
	`OR`

`{ }`	A set of choices in a syntax line.
`[ ]`	Optional items in a syntax line. *Example*: `java utils.MulticastTest -n name -a address [-p portnumber] [-t timeout] [-s send]`
`|`	Separates mutually exclusive choices in a syntax line. *Example*: `java weblogic.deploy [list|deploy|undeploy|update] password {application} {source}`
`...`	Indicates one of the following in a command line: ■ An argument can be repeated several times in the command line. ■ The statement omits additional optional arguments. ■ You can enter additional parameters, values, or other information

Convention	Usage
. . .	Indicates the omission of items from a code example or from a syntax line.

CHAPTER

1 weblogic.Admin Command-Line Reference

The `weblogic.Admin` utility is a command-line interface that you can use to administer, configure, and monitor WebLogic Server.

Like the Administration Console, this utility assumes the role of client that invokes administrative operations on the Administration Server, which is the central management point for all servers in a domain. (All Managed Servers retrieve configuration data from the Administration Server, and the Administration Server can access runtime data from all Managed Servers.) While the Administration Console interacts only with the Administration Server, the `weblogic.Admin` utility can access the Administration Server as well as all active server instances directly. If the Administration Server is down, you can still use the `weblogic.Admin` utility to retrieve runtime information from Managed Servers and invoke some administrative commands. However, you can save configuration changes to the domain's `config.xml` file only when you access the Administration Server.

To automate administrative tasks, you can invoke the `weblogic.Admin` utility from shell scripts. If you plan to invoke this utility multiple from a shell script, consider using the `BATCHUPDATE` command, which is described in "Running Commands in Batch Mode" on page 1-76.

The following sections describe using the `weblogic.Admin` utility:

Required Environment and Syntax for the weblogic.Admin Utility

Before you use the `weblogic.Admin` utility, set up your environment and note command syntax information as described in the following sections.

Environment

To set up your environment for the `weblogic.Admin` utility:

1. Install and configure the WebLogic Server software, as described in the *WebLogic Server Installation Guide*. See `http://e-docs.bea.com/wls/docs81b/install/index.html`.
2. Add WebLogic Server classes to the `CLASSPATH` environment variable. See "Setting the Classpath" on page 3-2.
3. If you want the `weblogic.Admin` utility to use a listen port that is reserved for administration traffic, you must configure a domain-wide administration port as described in "Enabling the Domain-Wide Administration Port" in the Administration Console Online Help.

   The domain-wide administration port is secured by SSL. For information about using secured ports with the `weblogic.Admin` utility, refer to "Java Options for SSL Communication" on page 1-3.

Note: If a server instance is deadlocked, it can respond to `weblogic.Admin` commands only if you have enabled the domain-wide administration port. If you have not already enabled the domain-wide administration port, your only option is to shut down the server instance by killing the Java process that is running the server. You will lose all session data. For information on enabling the domain-wide administration port, refer to "Enabling the Domain-Wide Administration Port" in the Administration Console Online Help.

Syntax

```
java [-Dweblogic.security.SSL.ignoreHostnameVerify=true]
     [-Dweblogic.security.TrustKeyStore=DemoTrust]
     weblogic.Admin [ [–url | -adminurl] [protocol://]listen-address:port]
     -username username -password password
     COMMAND-NAME arguments
```

The command names and arguments are not case sensitive.

The following sections provide additional syntax information:

- “Java Options for SSL Communication” on page 1-3
- “Protocol Support” on page 1-4
- “Common Arguments” on page 1-5

Java Options for SSL Communication

If you connect to a server instance through a secured port (such as the domain-wide administration port), note the following:

- You must specify a secure protocol in the `-url` or `-adminurl` argument. For example, `-url t3s://listen-address:secure-port`.
- If the server is using the default demonstration key stores, include the `-Dweblogic.security.TrustKeyStore=DemoTrust` Java option.
- If the server is using something other than the default demonstration key stores and if the server has specified an IP address as its Listen Address, include the `-Dweblogic.security.SSL.ignoreHostnameVerify=true` Java option.

Protocol Support

The `-url` and `-adminurl` arguments of the `weblogic.Admin` utility support the `t3`, `t3s`, `http`, and `https` protocols.

If you want to use `http` or `https` to connect to a server instance, you must enable HTTP Tunneling for that instance. For more information, refer to "Configuring the HTTP Protocol" in the Administration Console Online Help.

If you use the `-url` argument to specify a non-secured port, the `weblogic.Admin` utility uses `t3` by default. For example, `java weblogic.Admin -url localhost:7001` resolves to `java weblogic.Admin -url t3://localhost:7001`.

If you use either the `-url` or `-adminurl` argument to specify a port that is secured by SSL, you must specify either `t3s` or `https`. For example, if you enable the default SSL listen port, you can use the following URLs: `-url t3s://MyHost:7002` or `-url https://MyHost:7002`.

Common Arguments

Table 1-1 describes arguments that are common to most commands.

Table 1-1 Common Arguments

Argument	Definition
`-url [protocol://]listen-address:listen-port`	The listen address and listen port of the server instance that runs the command. In most cases, you should specify the Administration Server's address and port, which is the central management point for all servers in a domain. Some commands, such as START, CREATE_POOL, and CREATE, **must** run on the Administration Server. The documentation for each command indicates whether this is so. If you specify a Managed Server's listen address and port, the command can access data only for that server instance; you cannot run a command on one Managed Server to view or change data for another server instance. When you use MBean-related commands, you must specify the Administration Server's listen address and port to access Administration MBeans. To access Local Configuration MBeans or Runtime MBeans, you can specify the server instance on which the MBeans reside. (However, the `-adminurl` argument can also retrieve Local Configuration MBeans or Runtime MBeans from any server.) For more information on where MBeans reside, refer to "WebLogic Server Managed Resources and MBeans" in the *Programming WebLogic Management Services with JMX* guide. To use a listen port that is not secured by SSL, the format is `-url [protocol://]listen-address:port` To use a port that is secured by SSL, the format is `-url secure-protocol://listen-address:port` If you have set up a domain-wide administration port, you must specify the administration port number: `-url secure-protocol://listen-address:domain-wide-admin-port` For information about valid values for *protocol* and *secure-protocol*, refer to "Protocol Support" on page 1-4. For more information about the listen address and listen ports, refer to "-Dweblogic.ListenAddress=host" on page 3-10 and "-Dweblogic.ListenPort=portnumber" on page 3-10. For more information about the domain-wide administration port, refer to "Enabling the Domain-Wide Administration Port" in the Administration Console Online Help. The default value for this argument is `t3://localhost:7001`.

Table 1-1 Common Arguments

Argument	Definition
`-adminurl [protocol://]Admin-Server-listen-address:listen-port`	Enables the Administration Server to retrieve Local Configuration MBeans or Runtime MBeans for any server instance in the domain. For information about types of MBeans, refer to "WebLogic Server Managed Resources and MBeans" in the *Programming WebLogic Management Services with JMX* guide. For all commands other than the MBean commands, `-adminurl admin-address` and `-url admin-address` are synonymous. **Note:** If you use the `-url` argument to specify the Administration Server (instead of using the `-adminurl` argument), you can retrieve only the Local Configuration MBeans and Runtime MBeans for the Administration Server itself. The `-adminurl` value must specify the listen address and listen port of the Administration Server. To use a port that is not secured by SSL, the format is `-adminurl [protocol]Admin-Server-listen-address:port`. To use a port that is secured by SSL, the format is `-adminurl secure-protocol://Admin-Server-listen-address:port` If you have set up a domain-wide administration port, you must specify the administration port number: `-adminurl secure-protocol://Admin-Server-listen-address:domain-wide-admin-port` For information about valid values for *protocol* and *secure-protocol*, refer to "Protocol Support" on page 1-4. There is no default value for this argument.
`-username username`	The name of the user who is issuing the command. This user must have appropriate permission to view or modify the target of the command. For information about permissions for system administration tasks, refer to "Protecting System Administration Operations" in the *Configuring and Managing WebLogic Server* guide.
`-password password`	The password that is associated with the username.

Example Environment

In many of the examples throughout the sections that follow, it is assumed that a certain environment has been set up:

- The WebLogic Server administration domain is named MedRec.
- The Administration Server is named MedRecServer and listens on port 7001.
- The Administration Server uses the name of its host machine, AdminHost, as its listen address. For more information about the listen address and listen ports, refer to "-Dweblogic.ListenAddress=host" on page 3-10 and "-Dweblogic.ListenPort= portnumber" on page 3-10.
- The `weblogic` username has system-administrator privileges and uses `weblogic` for a password.
- A Managed Server named MedRecManagedServer uses the name of its host machine, ManagedHost, as its listen address and 8001 as its listen port.

Exit Codes Returned by weblogic.Admin

All `weblogic.Admin` commands return an exit code of `0` if the command succeeds and an exit code of `1` if the command fails.

To view the exit code from a Windows command prompt, enter `echo %ERRORLEVEL%` after you run a `weblogic.Admin` command. To view the exit code in a BASH shell, enter `echo $?`.

For example:

```
D:\>java weblogic.Admin -username weblogic -password weblogic GET -pretty -mbean
"MedRec:Name=MyServer,Type=Server" -property ListenPort

---------------------------
MBeanName: "MedRec:Name=MyServer,Type=Server"
        ListenPort: 7010

D:\>echo %ERRORLEVEL%
0
```

Commands for Managing the Server Life Cycle

Table 1-2 is an overview of commands that manage the life cycle of a server instance. Subsequent sections describe command syntax and arguments, and provide an example for each command. For more information about the life cycle of a server instance, refer to "Server Life Cycle" in the *Configuring and Managing WebLogic Server* guide.

Table 1-2 Overview of Commands for Managing the Server Life Cycle

Command	Description
`CANCEL_SHUTDOWN`	(Deprecated) Cancels the `SHUTDOWN` command for the WebLogic Server that is specified in the URL. See "CANCEL_SHUTDOWN" on page 1-10.
`DISCOVERMANAGEDSERVER`	Causes the Administration Server to re-establish its administrative control over Managed servers. See "DISCOVERMANAGEDSERVER" on page 1-11.
`FORCESHUTDOWN`	Immediately terminates a WebLogic Server process. See "FORCESHUTDOWN" on page 1-13.
`LOCK`	(Deprecated) Locks a WebLogic Server against non-privileged logins. Any subsequent login attempt initiates a security exception which may contain an optional string message. See "LOCK" on page 1-15.
`RESUME`	Makes a server available to receive requests from external clients. See "RESUME" on page 1-17.
`SHUTDOWN`	Shuts down a WebLogic Server. See "SHUTDOWN" on page 1-18.
`START`	Uses a configured Node Manager to start a Managed Server in the `RUNNING` state. See "START" on page 1-21.

Table 1-2 Overview of Commands for Managing the Server Life Cycle (Continued)

Command	Description
`STARTINSTANDBY`	Uses a configured Node Manager to start a Managed Server and place it in the `STANDBY` state. See "STARTINSTANDBY" on page 1-23.
`UNLOCK`	(Deprecated) Unlocks the specified WebLogic Server after a `LOCK` operation. See "UNLOCK" on page 1-26.

CANCEL_SHUTDOWN

(Deprecated) The `CANCEL_SHUTDOWN` command cancels the `SHUTDOWN` command for a specified WebLogic Server.

When you use the `SHUTDOWN` command, you can specify a delay (in seconds). An administrator may cancel the shutdown command during the delay period. Be aware that the SHUTDOWN command disables logins, and they remain disabled even after cancelling the shutdown. Use the `UNLOCK` command to re-enable logins.

See "SHUTDOWN" on page 1-18 and "UNLOCK" on page 1-26.

This command is deprecated because the ability to specify a delay in the `SHUTDOWN` command is also deprecated. Instead of specifying a delay in the `SHUTDOWN` command, you can now set attributes to control how a server shuts down. For more information, refer to "Controlling Graceful Shutdowns" and "Setting the Timeout Period for Forced Shutdown Operations" in the Administration Console Online Help.

Syntax

```
java weblogic.Admin [-url URL]
    -username username -password password
    CANCEL_SHUTDOWN
```

Example

The following example cancels the shutdown of a WebLogic Server instance that runs on a machine named ManagedHost and listens on port 8001:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
   -password weblogic CANCEL_SHUTDOWN
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

DISCOVERMANAGEDSERVER

Causes the Administration Server to re-establish administrative control over Managed Servers.

If the Administration Server fails while Managed Servers continue to run, or if you shut down the Administration Server while Managed Servers continue to run, you lose the ability to change the configuration or deploy modules to any server in the domain. To regain this administrative ability, you must restart the Administration Server. If you start an Administration Server in Production Mode, during its startup cycle it finds the last known set of Managed Servers and re-establishes a connection. For more information about discovering Managed Servers during an Administration Server's startup cycle, refer to the `-Dweblogic.management.discover` entry in Table 3-3 on page 9.

If the Administration Server is unable to automatically re-establish a connection to one or more Managed Servers during its startup cycle, you must use this command to re-establish administrative control.

The following situations can prevent the Administration Server from discovering a Managed Server:

- You start a server in Managed Server Independence mode while the Administration Server is down.
- A Managed Server's root directory is on a separate network partition from the Administration Server.
- A Managed Server is not in a `RUNNING` state when the Administration Server tries to reconnect. For example, you might have started the Managed Server in the `STANDBY` state and did not resume the Managed Server before restarting the Administration Server.

Other factors can prevent the Administration Server from finding and re-connecting to Managed Servers, and you can use this command any time you need to re-establish a connection.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   DISCOVERMANAGEDSERVER [-serverName targetServer]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`-serverName`	A Managed Server that is currently running. If you do not specify a server, the Administration Server will discover and re-establish control over all the Managed Servers that are known to be running but disconnected from administrative services

Example

The following command instructs the Administration Server to re-connect to a Managed Server:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic DISCOVERMANAGEDSERVER MedRecManagedServer
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

FORCESHUTDOWN

Immediately terminates a server instance.

When you initiate a forced shutdown, the server instructs subsystems to immediately drop in-work requests. For more information, refer to "Forced Shutdown" in the *Configuring and Managing WebLogic Server* guide.

If a server instance is in a deadlocked state, it can respond to `weblogic.Admin` commands only if you have enabled the domain-wide administration port. (A deadlocked server is one in which all threads are struck trying to acquire locks held by other threads.) If you have not already enabled the domain-wide administration port, your only option for shutting down the server instance is to kill the Java process that is running the server. You will lose all session data. For information on enabling the domain-wide administration port, refer to "Enabling the Domain-Wide Administration Port" in the Administration Console Online Help.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   FORCESHUTDOWN [targetServer]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the Administration Server. If the Administration Server is not available, specify the listen address and listen port of the server instance that you want to shut down and omit the *`targetServer`* argument. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`targetServer`	The name of the server to shut down. If you do not specify a value, the command shuts down the server that you specified in the `-url` argument.

Example

The following command instructs the Administration Server to shut down a Managed Server:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic FORCESHUTDOWN MedRecManagedServer
```

After you issue the command, MedRecManagedServer prints messages to its log file and to its standard out. The messages indicate that the server state is changing and that the shutdown sequence is starting.

If the command succeeds, the final message that the target server prints is as follows:

```
<Oct 12, 2002 11:28:59 AM EDT> <Alert> <WebLogicServer> <000219>
<The shutdown sequence has been initiated.>
```

In addition, if the command succeeds, the `weblogic.Admin` utility returns the following:

```
Server "MedRecManagedServer" was force shutdown successfully ...
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

In the following example, the Administration Server is not available, so the command instructs the Managed Server to shut itself down:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
-password weblogic FORCESHUTDOWN
```

LOCK

(Deprecated) Locks a WebLogic Server instance against non-privileged logins. Any subsequent login attempt initiates a security exception which may contain an optional string message.

Note: This command is privileged. It requires the password for the WebLogic Server administrative user.

Instead of using the `LOCK` command, start a server in the `STANDBY` state. In this state, a server instance responds only to administrative requests over the domain-wide administration port. For more information, refer to "STARTINSTANDBY" on page 1-23.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   LOCK ["stringMessage"]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the server instance that you want to lock. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
"stringMessage"	Message, in double quotes, to be supplied in the security exception that is thrown if a non-privileged user attempts to log in while the WebLogic Server instance is locked.

Example

In the following example, a Managed Server named MedRecManagedServer is locked.

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
   -password weblogic
   LOCK "Sorry, WebLogic Server is temporarily out of service."
```

Any application that subsequently tries to log into the locked server with a non-privileged username and password receives the specified message: `Sorry, WebLogic Server is temporarily out of service`.

RESUME

Moves a server instance from the STANDBY state to the RUNNING state.

For more information about server states, refer to "Server Life Cycle" in the *Configuring and Managing WebLogic Server* guide.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   RESUME [targetServer]
```

Argument	Definition
`-url` *`secure-protocol://listen-address:listen-port`*	Because servers can be in the STANDBY state only if the domain-wide administration port is enabled, to resume a server you must specify the Administration Server and domain-wide administration port as follows: `t3s://`*`Admin-Server-listen-address:domain-wide-admin-port`* or `https://`*`Admin-Server-listen-address:domain-wide-admin-port`* For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`targetServer`	The name of the server to resume. If you do not specify a value, the command resumes the server that you specified in the `-url` argument.

Example

The following example connects to the Administration Server and instructs it to resume a Managed Server:

```
java weblogic.Admin -url t3s://AdminHost:9002 -username weblogic
-password weblogic RESUME MedRecManagedServer
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

SHUTDOWN

Shuts down the specified WebLogic Server instance.

A graceful shutdown gives WebLogic Server subsystems time to complete certain application processing currently in progress. For information, refer to "Graceful Shutdown" in the *Configuring and Managing WebLogic Server* guide.

In release 6.x, this command included an option to specify a number of seconds to wait before starting the shutdown process. This option is now deprecated. To support this deprecated option, this command assumes that a numerical value in the field immediately after the `SHUTDOWN` command indicates seconds. Thus, you cannot use this command to gracefully shut down a server whose name is made up entirely of numbers. Instead, you must use the Administration Console. For information, refer to "Shutting Down a Server" in the Administration Console Online Help.

Instead of specifying a delay in the `SHUTDOWN` command, you can now set attributes to control how a server shuts down. For more information, refer to "Controlling Graceful Shutdowns" and "Setting the Timeout Period for Forced Shutdown Operations" in the Administration Console Online Help.

If a server instance is in a deadlocked state, it can respond to `weblogic.Admin` commands only if you have enabled the domain-wide administration port. (A deadlocked server is one in which all threads are struck trying to acquire locks held by other threads.) If you have not already enabled the domain-wide administration port, your only option for shutting down the server instance is to kill the Java process that is running the server. You will lose all session data. For information on enabling the domain-wide administration port, refer to "Enabling the Domain-Wide Administration Port" in the Administration Console Online Help.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   SHUTDOWN [-ignoreExistingSessions] [targetServer]

(Deprecated) java weblogic.Admin [-url URL]
   -username username -password password
   SHUTDOWN [-timeout seconds  [targetServer]] |
   [seconds [“stringMessage”]]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the Administration Server. If the Administration Server is not available, specify the listen address and listen port of the server instance that you want to shut down and omit the *`targetServer`* argument. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`-ignoreExistingSessions`	Causes a graceful shutdown operation to drop all HTTP sessions immediately. If you do not specify this option, the command refers to the Ignore Sessions During Shutdown setting for the server in the domain's `config.xml` file. For more information, refer to "Controlling Graceful Shutdowns" in the Administration Console Online Help. By default, a graceful shutdown operation waits for HTTP sessions to complete or timeout.
`targetServer`	The name of the server to shut down. If you do not specify a value, the command shuts down the server that you specified in the `-url` argument.
`-timeout` *`seconds`*	(Deprecated) Number of seconds allowed to elapse between the invoking of this command and the shutdown of the server.
`"stringMessage"`	(Deprecated) Message, in double quotes, to be supplied in the message that is sent if a user tries to log in while the WebLogic Server is being shut down.

Example

The following example instructs the Administration Server to shut down a Managed Server:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
-password weblogic SHUTDOWN MedRecManagedServer
```

After you issue the command, MedRecManagedServer prints messages to its log file and to its standard out. The messages indicate that the server state is changing and that the shutdown sequence is starting.

If the command succeeds, the final message that the target server prints is as follows:

```
<Oct 12, 2002 11:28:59 AM EDT> <Alert> <WebLogicServer> <000219>
<The shutdown sequence has been initiated.>
```

In addition, if the command succeeds, the `weblogic.Admin` utility returns the following:

```
Server "MedRecManagedServer" was shutdown successfully ...
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

In the following example, the Administration Server is not available. The same user connects to a Managed Server and instructs it to shut itself down:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
-password weblogic SHUTDOWN
```

START

Starts a Managed Server using Node Manager.

This command requires the following environment:

- The domain's Administration Server must be running.
- The Node Manager must be running on the Managed Server's host machine.
- The Managed Server's startup items and Node Manager settings must be set up as described in "Managing Server Availability with Node Manager" in the *Configuring and Managing WebLogic Server* guide.

Note: In the Administration Console, the Servers—General tab includes a Startup Mode field that you use to specify the state in which a server starts. However, this setting only applies if you start a server using the `weblogic.Server` command. The Node Manager, and therefore the `weblogic.Admin START` command, does not use the value that you specify. For example, even if you specify `STANDBY` as the value for the Startup Mode, if you issue the `weblogic.Admin START` command, the server will start in the `RUNNING` state.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   START targetServer
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Must specify the listen address and listen port of the domain's Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
targetServer	The name of the Managed Server to start in a `RUNNING` state.

Example

The following example instructs the Administration Server and Node Manager to start a Managed Server:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
-password weblogic START MedRecManagedServer
```

When you issue the command, the following occurs:

1. The Administration Server determines which machine `MedRecManagedServer` is configured to run on. It instructs the Node Manager that is running on that machine to start `MedRecManagedServer`.
2. The Node Manager indicates its progress by writing messages to its standard out. You can view these messages from the Administration Console on the Server—Control—Remote Start Output tab.
3. If the command succeeds, the `weblogic.Admin` utility returns to the following message:

```
Server "MedRecManagedServer" was started ...
Please refer to server log files for completion status ...
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

STARTINSTANDBY

Starts a Managed Server using Node Manager and places it in a `STANDBY` state. In this state, a server is not accessible to requests from external clients.

This command requires the following environment:

- The domain's Administration Server must be running.
- The Node Manager must be running on the Managed Server's host machine.
- The Managed Server's startup items and Node Manager settings must be set up as described in "Managing Server Availability with Node Manager" in the *Configuring and Managing WebLogic Server* guide.
- The domain must be configured to use a domain-wide administration port as described in "Enabling the Domain-Wide Administration Port" in the Administration Console Online Help.

For more information about server states, refer to "Server Life Cycle" in the *Configuring and Managing WebLogic Server* guide.

Note: In the Administration Console, the Servers—General tab includes a Startup Mode field that you use to specify the state in which a server starts. However, this setting only applies if you start a server using the `weblogic.Server` command. The Node Manager, and therefore the `weblogic.Admin STARTINSTANDBY` command, does not use the value that you specify. For example, even if you specify `RUNNING` as the value for the Startup Mode, if you issue the `weblogic.Admin STARTINSTANDBY` command, the server will start in the `STANDBY` state.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   STARTINSTANDBY targetServer
```

Argument	Definition
`-url` *`secure-protocol://listen-address:listen-port`*	You must specify the Administration Server and domain-wide administration port as follows: `t3s://`*`Admin-Server-listen-address:domain-wide-admin-port`* or `https://`*`Admin-Server-listen-address:domain-wide-admin-port`* For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`targetServer`	The name of the WebLogic Server to start in the `STANDBY` state.

Example

The following example instructs the Administration Server and Node Manager to start a Managed Server in a `STANDBY` state:

```
java weblogic.Admin -url t3s://AdminHost:9002 -username weblogic
-password weblogic STARTINSTANDBY MedRecManagedServer
```

When you issue the command, the following occurs:

1. The Administration Server determines which machine `MedRecManagedServer` is configured to run on. It instructs the Node Manager that is running on that machine to start `MedRecManagedServer`.
2. The Node Manager indicates its progress by writing messages to its standard out. You can view these messages from the Administration Console on the Server→Control→Remote Start Output tab.
3. If the command succeeds, the `weblogic.Admin` utility returns to the following message:

```
Server "MedRecManagedServer" was started ...
Please refer to server log files for completion status ...
```

When you use the Node Manager to start a Managed Server, the Node Manager writes standard out and standard error messages to its log file. You can view these messages from the Administration Console on the Machine→Monitoring tab.

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

UNLOCK

(Deprecated) Unlocks the specified WebLogic Server after a `LOCK` operation.

This command is deprecated because the `LOCK` command is deprecated. Instead of `LOCK` and `UNLOCK`, use `STARTINSTANDY` and `RESUME`. For more information, refer to "RESUME" on page 1-17.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   UNLOCK
```

Example

In the following example, an administrator named `adminuser` with a password of `gumby1234` requests the unlocking of the WebLogic Server listening on port 7001 on machine `localhost`:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 UNLOCK
```

Commands for Retrieving Information about WebLogic Server and Server Instances

Table 1-3 is an overview of commands that return information about WebLogic Server installations and instances of WebLogic Server. Subsequent sections describe command syntax and arguments, and provide an example for each command.

Table 1-3 Overview of Commands for Retrieving Information about WebLogic Server

Command	Description
`CONNECT`	Makes the specified number of connections to a WebLogic Server instance and returns two numbers representing the total time for each round trip and the average amount of time (in milliseconds) that each connection is maintained. See "CONNECT" on page 1-28.
`GETSTATE`	Returns the current state of the specified WebLogic Server instance. See "GETSTATE" on page 1-30.
`HELP`	Provides syntax and usage information for all WebLogic Server commands (by default) or for a single command if a command value is specified on the HELP command line. See "HELP" on page 1-32.
`LICENSES`	Lists the licenses for all WebLogic Server instances that are installed on a specific server. See "LICENSES" on page 1-33.
`LIST`	Lists the bindings of a node in a server's JNDI naming tree. See "LIST" on page 1-34.
`PING`	Sends a message to verify that a WebLogic Server instance is listening on a port and is ready to accept client requests. See "PING" on page 1-36. For a similar command that returns information about all servers in a cluster, see "CLUSTERSTATE" on page 1-80.
`SERVERLOG`	Displays the server log file generated on a specific server instance. See "SERVERLOG" on page 1-38.
`THREAD_DUMP`	Provides a real-time snapshot of the WebLogic Server threads that are currently running on a particular instance. See "THREAD_DUMP" on page 1-41.
`VERSION`	Displays the version of the WebLogic Server software that is running on the machine specified by the value of *`URL`*. See "VERSION" on page 1-43.

CONNECT

Connects to a WebLogic Server instance and returns two numbers representing the total time for each round trip and the average amount of time (in milliseconds) that each connection is maintained.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   CONNECT [count]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the server instance to which you want to connect. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
count	Number of connections the `weblogic.Admin` utility makes to the specified server instance. By default, this command makes only one connection.

Example

In the following example, the `weblogic.Admin` utility establishes 10 connections to a WebLogic Server instance whose listen address is ManagedHost and listen port is 8001:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
   -password weblogic CONNECT 10
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command establishes the connections, it returns the following information:

```
Connection: 0 - 3,229 ms
Connection: 1 - 17 ms
Connection: 2 - 14 ms
Connection: 3 - 20 ms
Connection: 4 - 18 ms
Connection: 5 - 25 ms
Connection: 6 - 27 ms
Connection: 7 - 15 ms
Connection: 8 - 15 ms
Connection: 9 - 15 ms
  RTT = ~3422 milliseconds, or ~342 milliseconds/connection
```

If the command does not establish a connection, it returns nothing.

In this example, the first connection required 3,229 milliseconds and the second connection required 17 milliseconds. The average time for all connections was 3422 milliseconds.

GETSTATE

Returns the current state of a server.

For more information about server states, refer to "Server Life Cycle" in the *Configuring and Managing WebLogic Server* guide.

If a server instance is in a deadlocked state, it can respond to `weblogic.Admin` commands only if you have enabled the domain-wide administration port. (A deadlocked server is one in which all threads are struck trying to acquire locks held by other threads.) If you have not already enabled the domain-wide administration port, your only option is to shut down the server instance by killing the Java process that is running the server. You will lose all session data. For information on enabling the domain-wide administration port, refer to "Enabling the Domain-Wide Administration Port" in the Administration Console Online Help.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   GETSTATE [targetServer]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the Administration Server. If the Administration Server is not available, specify the listen address and listen port of the server instance for which you want to retrieve the current state and omit the *`targetServer`* argument. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`targetServer`	The name of the server for which you want to retrieve the current state. If you do not specify a value, the command returns the state of the server that you specified in the `-url` argument.

Example

The following example returns the state of a WebLogic Server instance that runs on a machine named AdminHost:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic GETSTATE
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds for a running server, it returns the following:

```
Current state of "MedRecServer" : RUNNING
```

For a complete list of server states, refer to "Server Life Cycle" in the *Configuring and Managing WebLogic Server* guide.

HELP

Provides syntax and usage information for all WebLogic Server commands (by default) or for a single command if a command value is specified on the HELP command line.

You can issue this command from any computer on which the WebLogic Server is installed. You do not need to start a server instance to invoke this command, nor do you need to supply user credentials.

Syntax

```
java weblogic.Admin HELP [COMMAND]
```

Example

In the following example, information about using the `PING` command is requested:

```
java weblogic.Admin HELP PING
```

The command returns the following:

```
Usage: java [SSL trust options]
    weblogic.Admin [ [-url | -adminurl] [<protocol>://]<listen-address>:<port>]
    -username <username> -password <password>
    PING <roundTrips> <messageLength>

Where:
roundTrips = Number of pings.
messageLength = Size of the packet (in bytes) to send in each ping. The default
size is 100 bytes. Requests for pings with packets larger than 10 MB throw
exceptions.

Description: Sends a message to verify that a WebLogic Server instance is listen
ing on a port and is ready to accept WebLogic client requests.

Example(s):
Connecting through a non-secured port:
java weblogic.Admin -url t3://localhost:7001 -username weblogic -password weblog
icping 3 100

Connecting through an SSL port of a server that uses the demonstration keys and
certificates:
|java -Dweblogic.security.TrustKeyStore=DemoTrust
weblogic.Admin -url t3s:\localhost:7001 -username weblogic -password weblogic
PING <roundTrips> <messageLength>
```

LICENSES

Lists the BEA licenses for all WebLogic Server instances installed on the specified host.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   LICENSES
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of a WebLogic Server instance. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.

Example

The following command returns a list of licenses for a host named AdminHost:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic LICENSES
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command establishes a connection, it returns license information to standard out.

LIST

Lists the bindings of a node in the JNDI naming tree.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   LIST [JNDIcontextName]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the server instance for which you want to retrieve the JNDI naming tree. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
JNDIcontextName	The JNDI context for lookup, for example, `weblogic`, `weblogic.ejb`, `javax`. By default, the command lists the bindings immediately below the InitialContext of the specified server instance.

Example

The following command returns the initial context for server instance that runs on a machine named AdminHost:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic LIST
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it returns information similar to the following.

```
Contents of InitialContext
  javax: weblogic.jndi.internal.ServerNamingNode
  mail: weblogic.jndi.internal.ServerNamingNode
  weblogic: weblogic.jndi.internal.ServerNamingNode
```

```
mqseries: weblogic.jndi.internal.ServerNamingNode
jms: weblogic.jndi.internal.ServerNamingNode
MedRecTxDataSource: weblogic.jdbc.common.internal.RmiDataSource
MedRecDataSource: weblogic.jdbc.common.internal.RmiDataSource
```

The following command returns the JNDI bindings for the `mail` context:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic LIST mail
```

If the command succeeds, it returns the following:

```
Contents of mail
  MedRecMailSession: javax.mail.Session
```

PING

Sends a message to verify that a WebLogic Server instance is listening on a port and is ready to accept WebLogic client requests.

For information on returning a description of all servers in a cluster, refer to "CLUSTERSTATE" on page 1-80.

If a server instance is in a deadlocked state, it can respond to `weblogic.Admin` commands only if you have enabled the domain-wide administration port. (A deadlocked server is one in which all threads are struck trying to acquire locks held by other threads.) If you have not already enabled the domain-wide administration port, your only option is to shut down the server instance by killing the Java process that is running the server. You will lose all session data. For information on enabling the domain-wide administration port, refer to "Enabling the Domain-Wide Administration Port" in the Administration Console Online Help.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   PING [roundTrips] [messageLength]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the server instance you want to ping. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`roundTrips`	Number of pings.
`messageLength`	Size of the packet (in bytes) to be sent in each ping. Requests for pings with packets larger than 10 MB throw exceptions.

Example

The following command pings a server instance 10 times:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic PING 10
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it returns output similar to the following:

```
Sending 10 pings of 100 bytes.
  RTT = ~46 milliseconds, or ~4 milliseconds/packet
```

The following command pings a server instance that is running on a host computer named ManagedHost:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
   -password weblogic PING
```

SERVERLOG

Returns messages from the local log file of a server instance. The command returns messages only from the current log file; it does not return messages in log files that the server instance has archived (renamed) because of log file rotation.

This command can not be used to return the domain-wide log file. You can view the domain-wide log file from the Administration Console. For more information about server log files, refer to "Local Log Files and Domain Log Files" in the Administration Console Online Help.

If you omit the `starttime` and `endtime` arguments, the command returns all log messages in the current log file up to a maximum of 500 messages.

For each message, the command returns the following message attributes, separated by spaces:

```
MessageID TimeStamp Severity Subsystem MessageText
```

For more information about message attributes, refer to "Message Attributes" in the Administration Console Online Help.

Syntax

```
java.weblogic.Admin [-url URL]
   -username username -password password
   SERVERLOG [starttime [endtime]]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the server instance for which you want to retrieve the local log file. If you use the `-url` argument to specify the Administration Server, the command returns the local log file of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.

Argument	Definition
starttime	Returns only the messages in the current log file with a time stamp that is **after** the time you specify. The date format is *yyyy/mm/dd*. Time is indicated using a 24-hour clock. The start date and time are entered inside quotation marks, in the following format: "*yyyy/mm/dd hh:mm*" By default, SERVERLOG returns all messages in chronological order starting from the beginning of the current log file.
endtime	Specifies the end of a time range and causes SERVERLOG to return only the messages with a time stamp that is after *starttime* and before *endtime*. The date format is *yyyy/mm/dd*. Time is indicated using a 24-hour clock. The end date and time are entered inside quotation marks, in the following format: "*yyyy/mm/dd hh:mm*" By default, SERVERLOG returns up to 500 messages in chronological starting with the *starttime* value and ending with the time at which you issued the SERVERLOG command.

Example

The following command returns all messages in the local log file of a server instance named MedRecManagedServer and pipes the output through the command shell's more command:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
   -password weblogic SERVERLOG | more
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it returns output similar to the following truncated example:

```
130036    Oct 18, 2002 4:19:12 PM EDT Info  XML   Initializing XMLRegistry.
001007    Oct 18, 2002 4:19:13 PM EDT  Info   JDBC  Initializing... issued.
001007    Oct 18, 2002 4:19:13 PM EDT  Info JDBC   Initialize Done issued.
190000    Oct 18, 2002 4:19:13 PM EDT   Info   Connector  Initializing J2EE
Connector Service
190001   Oct 18, 2002 4:19:13 PM EDT    Info    Connector J2EE Connector Service
initialized successfully
...
```

The following command returns messages that were written to the local log file since 8:00 am today:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
   -password weblogic SERVERLOG 08:00
```

The following command returns messages that were written to the local log file between 8:00 am and 8:30 am on October 18, 2002:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
 -password weblogic SERVERLOG "2002/10/18 08:00" "2002/10/18 08:30"
```

THREAD_DUMP

Prints a snapshot of the WebLogic Server threads that are currently running for a specific server instance. The server instance prints the snapshot to its standard out.

If a server instance is in a deadlocked state, it can respond to `weblogic.Admin` commands only if you have enabled the domain-wide administration port. (A deadlocked server is one in which all threads are struck trying to acquire locks held by other threads.) If you have not already enabled the domain-wide administration port, your only option is to shut down the server instance by killing the Java process that is running the server. You will lose all session data. For information on enabling the domain-wide administration port, refer to "Enabling the Domain-Wide Administration Port" in the Administration Console Online Help.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password THREAD_DUMP
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the server instance for which you want to view the thread dump. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.

Example

The following example causes a server instance that is running on a host named ManagedHost to print a thread dump to standard out:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
-password weblogic THREAD_DUMP
```

If the command succeeds, the command itself returns the following:

```
Thread Dump is available in the command window that is running the
server.
```

The server instance prints a thread dump to its standard out, which, by default, is the shell (command prompt) within which the server instance is running.

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

VERSION

Displays the version of the WebLogic Server software that is running the server instance you specify with the `-url` argument.

Syntax

```
java weblogic.Admin [-url URL] -username username
   -password password VERSION
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of a WebLogic Server instance. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.

Example

The following command displays the version of the WebLogic Server software that is currently running on a host named ManagedHost:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
   -password weblogic VERSION
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it returns output similar to the following:

```
WebLogic Server 8.1  Sat Oct 15 22:51:04 EDT 2002 207896
WebLogic XMLX Module 8.1  Sat Oct 15 22:51:04 EDT 2002 207896
```

Commands for Managing JDBC Connection Pools

Table 1-4 is an overview of WebLogic Server administration commands for connection pools. Subsequent sections describe command syntax and arguments, and provide an example for each command.

For additional information about connection pools see *Programming WebLogic JDBC* at `http://e-docs.bea.com/wls/docs81b/jdbc/index.html` and "JDBC Connection Pools" in the Administration Console Online Help.

Table 1-4 Overview of Commands for Managing JDBC Connection Pools

Command	Description
`CREATE_POOL`	Allows creation of connection pool while WebLogic Server is running. Note that dynamically created connection pools cannot be used with DataSources or TxDataSources. See "CREATE_POOL" on page 1-46.
`DESTROY_POOL`	Connections are closed and removed from the pool and the pool dies when it has no remaining connections. See "DESTROY_POOL" on page 1-49.
`DISABLE_POOL`	You can temporarily disable a connection pool, preventing any clients from obtaining a connection from the pool. See "DISABLE_POOL" on page 1-50.
`ENABLE_POOL`	When a pool is enabled after it has been disabled, the JDBC connection states for each in-use connection are exactly as they were when the connection pool was disabled; clients can continue JDBC operations exactly where they left off. See "ENABLE_POOL" on page 1-52.
`EXISTS_POOL`	Tests whether a connection pool with a specified name exists in a WebLogic Server instance. Use this command to determine whether a dynamic connection pool has already been created or to ensure that you select a unique name for a dynamic connection pool you want to create. See "EXISTS_POOL" on page 1-53.

Table 1-4 Overview of Commands for Managing JDBC Connection Pools

Command	Description
`RESET_POOL`	Closes and reopens all allocated connections in a connection pool. This may be necessary after the DBMS has been restarted, for example. Often when one connection in a connection pool has failed, all of the connections in the pool are bad. See "RESET_POOL" on page 1-54.

CREATE_POOL

Allows creation of connection pool while WebLogic Server is running. For more information, see "Creating a Connection Pool Dynamically" in *Programming WebLogic JDBC* at `http://e-docs.bea.com/wls/docs81b/jdbc/programming.html#programming004`.

Note that dynamically created connection pools cannot be used with DataSources or TxDataSources.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   CREATE_POOL poolName  aclName=aclX,
   props=myProps,initialCapacity=1,maxCapacity=1,
   capacityIncrement=1,allowShrinking=true,shrinkPeriodMins=15,
   driver=myDriver,url=myURL
```

Argument	Definition
`-url` [*protocol*`://`]*listen-address*`:`*listen-port*	Specify the listen address and listen port of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`poolName`	Required. Unique name of pool.
`aclName`	Required. Identifies the different access lists within `fileRealm.properties` in the server config directory. Paired name must be dynaPool.
`props`	Database connection properties; typically in the format "database login name; database password; server network id".
initialCapacity	Initial number of connections in a pool. If this property is defined and a positive number > 0, WebLogic Server creates these connections at boot time. Default is 1; cannot exceed maxCapacity.
maxCapacity	Maximum number of connections allowed in the pool. Default is 1; if defined, maxCapacity should be =>1.

Argument	Definition
capacityIncrement	Number of connections that can be added at one time. Default = 1.
allowShrinking	Indicates whether or not the pool can shrink when connections are detected to not be in use. Default = true.
shrinkPeriodMins	Required. Interval between shrinking. Units in minutes. Minimum = 1.If `allowShrinking = True`, then default = 15 minutes.
driver	Required. Name of JDBC driver. Only local (non-XA) drivers can participate.
url	Required. URL of the JDBC driver.
testConnsOnReserve	Indicates reserved test connections. Default = False.
testConnsOnRelease	Indicates test connections when they are released. Default = False.
testTableName	Database table used when testing connections; must be present for tests to succeed. Required if either testConnOnReserve or testConOnRelease are defined.
refreshPeriod	Sets the connection refresh interval. Every unused connection will be tested using TestTableName. Connections that do not pass the test will be closed and reopened in an attempt to reestablish a valid physical database connection. If TestTableName is not set then the test will not be performed.
loginDelaySecs	The number of seconds to delay before creating each physical database connection. This delay takes place both during initial pool creation and during the lifetime of the pool whenever a physical database connection is created. Some database servers cannot handle multiple requests for connections in rapid succession. This property allows you to build in a small delay to let the database server catch up. This delay takes place both during initial pool creation and during the lifetime of the pool whenever a physical database connection is created.

Example

In the following example, a user with the name `weblogic` and the password `weblogic` runs the `CREATE_POOL` command to create a dynamic connection pool:

```
java weblogic.Admin -url localhost:7001 -username weblogic
   -password weblogic CREATE_POOL myPool
```

```
java weblogic.Admin -url forest:7901 -username weblogic
   -password weblogic CREATE_POOL dynapool6 "aclName=someAcl,
   allowShrinking=true,shrinkPeriodMins=10,
   url=jdbc:weblogic:oracle,driver=weblogic.jdbc.oci.Driver,
    initialCapacity=2,maxCapacity=8,
    props=user=SCOTT;password=tiger;server=bay816"
```

DESTROY_POOL

Connections are closed and removed from the pool and the pool dies when it has no remaining connections.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   DESTROY_POOL poolName [true|false]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
poolName	Required. Unique name of pool.
false (soft shutdown)	Soft shutdown waits for connections to be returned to the pool before closing them.
true (default—hard shutdown)	Hard shutdown kills all connections immediately. Clients using connections from the pool get exceptions if they attempt to use a connection after a hard shutdown.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `DESTROY_POOL` command temporarily freeze the active pool connections:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 DESTROY_POOL myPool false
```

DISABLE_POOL

You can temporarily disable a connection pool, preventing any clients from obtaining a connection from the pool.

You have to options for disabling a pool. 1) Freezing the connections in a pool that you later plan to enable, and 2) destroy the connections.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   DISABLE_POOL poolName [true|false]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
poolName	Name of the connection pool
false (disables and **suspends**)	Disables the connection pool, and suspends clients that currently have a connection. Attempts to communicate with the database server throw an exception. Clients can, however, close their connections while the connection pool is disabled; the connections are then returned to the pool and cannot be reserved by another client until the pool is enabled.
true (default—disables and **destroys**)	Disables the connection pool, and destroys the client's JDBC connection to the pool. Any transaction on the connection is rolled back and the connection is returned to the connection pool.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `DISABLE_POOL` command to freeze a connection that is to be enabled later:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 DISABLE_POOL myPool false
```

ENABLE_POOL

When a pool is enabled, the JDBC connection states for each in-use connection are exactly as they were when the connection pool was disabled; clients can continue JDBC operations exactly where they left off.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   ENABLE_POOL poolName
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
poolName	Name of the connection pool.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `ENABLE_POOL` command to reestablish connections that have been disabled (frozen):

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 ENABLE_POOL myPool
```

EXISTS_POOL

Tests whether a connection pool with a specified name exists in the WebLogic Server. You can use this method to determine whether a dynamic connection pool has already been created or to ensure that you select a unique name for a dynamic connection pool you want to create.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   EXISTS_POOL poolName
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
poolName	Name of connection pool.

Example

In the following example, a user with the name `adminuser` and the password `gumby1234` runs the `EXISTS_POOL` command to determine wether or not a pool with a specific name exists:

```
java weblogic.Admin -url localhost:7001 -username adminuser
   -password gumby1234 EXISTS_POOL myPool
```

RESET_POOL

This command resets the connections in a registered connection pool.

This is a privileged command. You must supply the password for the WebLogic Server administrative user to use this command. You must know the name of the connection pool, which is an entry in the `config.xml` file.

Syntax

```
java weblogic.Admin [-url URL]
  -username username -password password
  RESET_POOL poolName system password
```

Argument	Definition
URL	The URL of the WebLogic Server host and port number of the TCP port at which WebLogic is listening for client requests; use `"host:port."`
poolName	Name of a connection pool as it is registered in the WebLogic Server's config.xml file.
password	Password to be authenticated so commands can be executed. Default is the password that is associated with the default username.

Example

This command refreshes the connection pool registered as "eng" for the WebLogic Server listening on port 7001 of the host xyz.com.

```
java weblogic.Admin xyz.com:7001 RESET_POOL eng system gumby
```

TEST_POOL

Tests a connection pool by reserving and releasing a connection from it. If the pool is configured to test reserved connections or test released connections, this command also tests the reserve and release operations.

Syntax

```
java weblogic.Admin [-url URL]
  -username username -password password
  TEST_POOL poolName
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of a WebLogic Server instance on which the connection pool has been deployed. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
poolName	Name of a connection pool as it is registered in the WebLogic Server's config.xml file.

Example

This command tests the connection pool registered as `MedRecPool` and deployed on a server that listens on port 7001 of the host AdminHost:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
-password weblogic TEST_POOL MedRecPool
```

If the command succeeds, it returns the following:

```
JDBC Connection Test Succeeded for connection pool "MedRecPool".
```

Commands for Managing WebLogic Server MBeans

The following sections describe `weblogic.Admin` commands for managing WebLogic Server MBeans.

Specifying MBean Types

To specify which MBean or MBeans you want to access, view, or modify, all of the MBean management commands require either the `-mbean` argument or the `-type` argument.

Use the `-mbean` argument to operate on a single instance of an MBean.

Use the `-type` argument to operate on all MBeans that are an instance of a type that you specify. An MBean's **type** refers to the interface class of which the MBean is an instance. All WebLogic Server MBeans are an instance of one of the interface classes defined in the `weblogic.management.configuration` or `weblogic.management.runtime` packages. For configuration MBeans, type also refers to whether an instance is an Administration MBean or a Local Configuration MBean. For a complete list of all WebLogic Server MBean interface classes, refer to the WebLogic Server Javadoc for the `weblogic.management.configuration` or `weblogic.management.runtime` packages.

To determine the value that you provide for the `-type` argument, do the following:

1. Find the MBean's interface class and remove the `MBean` suffix from the class name. For an MBean that is an instance of the `weblogic.management.runtime.JDBCConnectionPoolRuntimeMBean`, use `JDBCConnectionPoolRuntime`.

2. For a Local Configuration MBean, append `Config` to the name. For example, for a Local Configuration MBean that is an instance of the `weblogic.management.configuration.JDBCConnectionPoolMBean` interface class, use `JDBCConnectionPoolConfig`. For the corresponding Administration MBean instance, use `JDBCConnectionPool`.

MBean Management Commands

Table 1-5 is an overview of the MBean management commands.

Table 1-5 MBean Management Command Overview

Command	Description
`CREATE`	Creates an Administration MBean instance. This command cannot be used for Runtime MBeans and we recommend that you do not use it to create Local Configuration MBeans. See "CREATE" on page 1-58.
`DELETE`	Deletes an MBean instance. See "DELETE" on page 1-61.
`GET`	Displays properties of MBeans. See "GET" on page 1-63.
`INVOKE`	Invokes management operations that an MBean exposes for its underlying resource. See "INVOKE" on page 1-66.
`QUERY`	Searches for MBeans whose `WebLogicObjectName` matches a pattern that you specify. See "QUERY" on page 68.
`SET`	Sets the specified property values for the named MBean instance. This command cannot be used for Runtime MBeans. See "SET" on page 1-71.

CREATE

Creates an instance of a WebLogic Server Administration or Local Configuration MBean, however, we recommend that you do not use it to create Local Configuration MBeans. This command cannot be used for Runtime MBeans.

If the command is successful, it returns `OK`.

When you use this command to create an Administration MBean instance, you must use the `-url` argument to specify the Administration Server. WebLogic Server populates the Administration MBean with default values and saves the MBean's configuration in the domain's `config.xml` file. For some types of Administration MBeans, WebLogic Server does not create the corresponding Local Configuration MBean replica until you restart the server instance that hosts the underlying managed resource. For example, if you create a `JDBCConnectionPool` Administration MBean to manage a JDBC connection pool on a Managed Server named ManagedMedRecServer, you must restart ManagedMedRecServer so that it can create its local replica of the `JDBCConnectionPool` Administration MBean that you created. For more information on MBean replication and the life cycle of MBeans, refer to "MBeans for Configuring Managed Resources" in the *Programming WebLogic Management Services* with JMX guide.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   CREATE –name name –type mbeanType
```

or

```
java weblogic.Admin [–url URL
   -username username -password password
   CREATE –mbean objectName
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specify the listen address and listen port of the Administration Server. You can create Administration MBeans only on the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4. Although the `CREATE` command also supports the `-adminurl` argument, we recommend that you do not use `CREATE` to create Local Configuration MBeans.
`-name name`	The name you choose for the MBean that you are creating.
`-type mbeanType`	The type of MBean that you are creating. For more information, refer to "Specifying MBean Types" on page 1-56.
`-mbean objectName`	Fully qualified object name of an MBean in the `WebLogicObjectName` format. For example: `"domain:Type=type,Name=name"` For more information, refer to the Javadoc for `WebLogicObjectName`.

Example

The following example uses the `-name` and `-type` arguments to create a `JDBCConnectionPool` Administration MBean named `myPool` on an Administration Server:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic CREATE -name myPool -type JDBCConnectionPool
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it prints the following to standard out:

```
Ok
```

The following example uses the `-mbean` argument and `WebLogicObjectName` conventions to create a `JDBCConnectionPool` Administration MBean named `myPool` on an Administration Server:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic
    CREATE -mbean “mydomain:Type=JDBCConnectionPool,Name=myPool”
```

DELETE

Deletes MBeans. If you delete an Administration MBean, WebLogic Server removes the corresponding entry from the domain's `config.xml` file.

If the command is successful, it returns `OK`.

Note: When you delete an Administration MBean, a WebLogic Server instance does not delete the corresponding Configuration MBean until you restart the server instance.

Syntax

```
java weblogic.Admin [ {-url URL} | {-adminurl URL} ]
   -username username -password password
   DELETE {–type mbeanType|–mbean objectName}
```

Arguments	Definition
`{-url [protocol://]listen-address:listen-port}` or `{-adminurl [protocol://]Admin-Server-listen-address:listen-port}`	To delete Administration MBeans, use `-url` to specify the Administration Server's listen address and listen port. To delete Runtime MBeans or Local Configuration MBeans, use one of the following: ■ `-url` to specify the listen address and listen port of the server instance on which you want to delete MBeans. ■ `-adminurl` to delete instances of a Runtime or Local Configuration MBean type from all server instances in the domain. For more information, refer to the `-url` and `-adminurl` entries in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`-type mbeanType`	Deletes all MBeans of the specified type. For more information, refer to "Specifying MBean Types" on page 1-56.
`-mbean objectName`	Fully qualified object name of an MBean in the `WebLogicObjectName` format. For example: `"domain:Type=type,Name=name"` For more information, refer to the Javadoc for `WebLogicObjectName`.

Example

The following example deletes the `JDBCConnectionPool` Administration MBean named `myPool`:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
 -password weblogic DELETE -mbean
 MedRec:Name=myPool,Type=JDBCConnectionPool
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it prints the following to standard out:

```
Ok
```

The following example deletes the `JDBCConnectionPool` Local Configuration MBean named `myPool` on a server instance named MedRecManagedServer:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
 -password weblogic DELETE -mbean
  MedRec:Location=MedRecManagedServer,Name=myPool,
 Type=JDBCConnectionPoolConfig
```

The following example deletes all `JDBCConnectionPool` Local Configuration MBeans for all server instances in the domain:

```
java weblogic.Admin -adminurl AdminHost:7001 -username weblogic
   -password weblogic DELETE -type JDBCConnectionPoolConfig
```

The following example deletes all `JDBCConnectionPool` Local Configuration MBeans on a server instance named `MedRecManagedServer`:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
 -password weblogic DELETE -type JDBCConnectionPoolConfig
```

GET

Displays MBean properties (attributes) and JMX object names (in the `WebLogicObjectName` format).

The output of the command is as follows:

```
{MBeanName object-name {property1 value} {property2 value}. . .}
{MBeanName object-name {property1 value} {property2 value} . . .}
. . .
```

Note that the properties and values are expressed as name-value pairs, each of which is returned within curly brackets. This format facilitates parsing of the output by a script.

If `-pretty` is specified, each property-value pair is displayed on a new line and curly brackets are not used to separate the pairs:

```
MBeanName: object-name
property1: value
property2: value
.
.
.
MBeanName: object-name
property1: value
abbribute2: value
```

Syntax

```
java weblogic.Admin [ {-url URL} | {-adminurl URL} ]
   -username username -password password
   GET [-pretty] {–type mbeanType|–mbean objectName}
   [–property property1] [–property property2]...
```

Argument	Definition
{-url [*protocol*://]*listen-address*:*listen-port*} or {-adminurl [*protocol*://]Admin-Server-*listen-address*:*listen-port*}	To retrieve Administration MBeans, use -url to specify the Administration Server's listen address and listen port. To retrieve Runtime MBeans or Local Configuration MBeans, use one of the following: ■ -url to specify the listen address and listen port of the server instance on which you want to retrieve MBeans. ■ -adminurl to retrieve instances of a Runtime or Local Configuration MBean type from all server instances in the domain. For more information, refer to the -url and -adminurl entries in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
-type *mbeanType*	Returns information for all MBeans of the specified type. For more information, refer to "Specifying MBean Types" on page 1-56.
-mbean *objectName*	Fully qualified object name of an MBean in the WebLogicObjectName format: *"domain*:Type=*type*,Location:*location*,Name=*name"* For more information, refer to the Javadoc for WebLogicObjectName.
-pretty	Places property-value pairs on separate lines.
-property *property*	The name of the MBean property (attribute) or properties to be listed. **Note:** If property is not specified using this argument, all properties are displayed.

Example

The following example displays all properties of the JDBCConnectionPool Administration MBean for a connection pool named MedRecPool. Note that the command must connect to the Administration Server to retrieve information from an Administration MBean:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
 -password weblogic GET -pretty -mbean
 MedRec:Name=MedRecPool,Type=JDBCConnectionPool
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it returns output similar to the following truncated example:

```
--------------------------
MBeanName: "MedRec:Name=MedRecPool,Type=JDBCConnectionPool"
        ACLName:
        CachingDisabled: true
        CapacityIncrement: 1
        ConnLeakProfilingEnabled: false
        ConnectionCreationRetryFrequencySeconds: 0
        ConnectionReserveTimeoutSeconds: 10
...
```

The following example displays all instances of all `JDBCConnectionPoolRuntime` MBeans for all servers in the domain.

```
java weblogic.Admin -adminurl AdminHost:7001 -username weblogic
 -password weblogic GET -pretty -type JDBCConnectionPoolRuntime
```

The following example displays all instances of all `JDBCConnectionPoolRuntime` MBeans that have been deployed on the server instance that listens on `ManagedHost:8001`:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
 -password weblogic GET -pretty -type JDBCConnectionPoolRuntime
```

INVOKE

Invokes a management operation for one or more MBeans. For WebLogic Server MBeans, you usually use this command to invoke operations other than the `get`*`Attribute`* and `set`*`Attribute`* that most WebLogic Server MBeans provide.

Syntax

```
java weblogic.Admin [ {-url URL} | {-adminurl URL} ]
   -username username -password password
   INVOKE {–type mbeanType|–mbean objectName} –method
   methodname [argument . . .]
```

Arguments	Definition

`{-url [`*`protocol`*`://]`*`listen-address`*`:`*`listen-port`*`}`
or
`{-adminurl [`*`protocol`*`://]Admin-Server-`*`listen-address`*`:`*`listen-port`*`}`
	To invoke operations for Administration MBeans, use `-url` to specify the Administration Server's listen address and listen port.
	To invoke operations for Runtime MBeans, use one of the following:
	■ `-url` to specify the listen address and listen port of the server instance on which you want to invoke Runtime MBean operations.
	■ `-adminurl` to invoke operations for all instances of a Runtime MBean on all server instances in the domain.
	We recommend that you do not invoke operations for Local Configuration MBeans. Instead, invoke the operation on the corresponding Administration MBean.

`-type` *`mbeanType`*	Invokes the operation on all MBeans of a specific type. For more information, refer to "Specifying MBean Types" on page 1-56.
`-mbean` *`objectName`*	Fully qualified object name of an MBean, in the `WebLogicObjectName` format: `"domain:Type=`*`type`*`,Location=`*`location`*`,Name=`*`name`*`"` For more information refer to the Javadoc for `WebLogicObjectName`.
`-method` *`methodname`*	Name of the method to be invoked.
`argument`	Arguments to be passed to the method call. When the argument is a String array, the arguments must be passed in the following format: *`"String1;String2;. . . "`*

Example

The following example enables a JDBC connection pool by invoking the `enable` method of the `JDBCConnectionPoolRuntime` MBean:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
 -password weblogic INVOKE
 -mbean MedRec:Location=MedRecServer,Name=myPool,
  ServerRuntime=MedRec,Type=JDBCConnectionPoolRuntime
 -method enable
```

If the command succeeds, it returns the following:

```
{MBeanName="MedRec:Location=MedRecServer,Name=MedRecPool,ServerRu
ntime=MedRecServer,Type=JDBCConnectionPoolRuntime"}

Ok
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

The following example enables all JDBC connection pools in the domain by invoking the `enable` method of all the `JDBCConnectionPoolRuntime` MBeans:

```
java weblogic.Admin -adminurl AdminHost:7001 -username weblogic
 -password weblogic INVOKE
 -type JDBCConnectionPoolRuntime -method enable
```

QUERY

Searches for WebLogic Server MBeans whose `WebLogicObjectName` matches a pattern that you specify.

All MBeans that are created from a WebLogic Server MBean type are registered in the MBean Server under a name that conforms to the `weblogic.management.WebLogicObjectName` conventions. You must know an MBean's `WebLogicObjectName` if you want to use `weblogic.Admin` commands to retrieve or modify specific MBean instances. For more information, refer to "WebLogicObjectNames for WebLogic Server MBeans" in the *Programming WebLogic Management Services with JMX* guide.

The output of the command is as follows:

```
{MBeanName object-name {property1 value} {property2 value}. . .}
{MBeanName object-name {property1 value} {property2 value} . . .}
. . .
```

Note that the properties and values are expressed as name-value pairs, each of which is returned within curly brackets. This format facilitates parsing of the output by a script.

If `-pretty` is specified, each property-value pair is displayed on a new line and curly brackets are not used to separate the pairs:

```
MBeanName: object-name
property1: value
property2: value
.
.
.
MBeanName: object-name
property1: value
abbribute2: value
```

Syntax

```
java weblogic.Admin [{-url URL} | {-adminurl URL}]
   -username username -password password
   QUERY -pretty –pattern object-name-pattern
```

Argument	Definition
{-url [*protocol*://]*listen-address*:*listen-port*} or {-adminurl [*protocol*://]Admin-Server-*listen-address*:*listen-port*}	To search for Administration MBean object names, use -url to specify the Administration Server's listen address and listen port. To search for the object names of Local Configuration or Runtime MBeans, use one of the following: ■ -url to specify the listen address and listen port of the server instance on which you want to search. ■ -adminurl to search on all server instances in the domain. For more information, refer to the -url and -adminurl entries in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
-pretty	Places property-value pairs on separate lines.
-pattern	A partial WebLogicObjectName for which the QUERY command searches. The value must conform to the following pattern: *domain-name*:*property-list* For the *domain-name* portion of the pattern, you can use the * character, which matches any character sequence. Because the server instance that you specify with the -url or -adminurl argument can access only the MBeans that belong to its domain, the * character is sufficient. For example, if you use -url to specify a server in the MedRec domain, QUERY can only return MBeans that are in the MedRec domain. It cannot search for MBeans in a domain named mydomain. For the *property-list* portion of the pattern, specify one or more components (property-value pairs) of a WebLogicObjectName. For a list of all WebLogicObjectName property-value pairs, refer to "WebLogicObjectNames for WebLogic Server MBeans" in the *Programming WebLogic Management Services with JMX* guide. (For example, all WebLogicObjectNames include Name=*value* and Type=*value* property-value pairs.) You can specify these property-value pairs in any order. Within a given naming property-value pair, there is no pattern matching. Only complete property-value pairs are used in pattern matching. However, you can use the * wildcard character in the place of one or more property-value pairs. For example, Name=Med* is not valid, but Name=MedRecServer,* is valid. If you provide at least one property-value pair in the *property-list*, you can locate the wildcard anywhere in the given pattern, provided that the *property-list* is still a comma-separated list.

Example

The following example searches for all `JDBCConnectionPoolRuntime` MBeans that are on a server instance that listens at ManagedHost:8001:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
 -password weblogic QUERY
 -pattern *:Type=JDBCConnectionPoolRuntime,*
```

If the command succeeds, it returns the following:

```
Ok
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

The following example searches for all instances of `MedRecPool` MBeans on all servers in the current domain. It uses `-adminurl`, which instructs the Administration Server to query the Administration `MBeanHome` interface (This interface has access to all MBeans in the domain):

```
java weblogic.Admin -adminurl AdminHost:7001 -username weblogic
 -password weblogic QUERY -pattern *:Name=MedRecPool,*
```

If the command succeeds, it returns an instance of the `JDBCConnectionPool` Administration MBean that is named `MedRecPool`, along with all corresponding Local Configuration and Runtime MBeans.

SET

Sets the specified property (attribute) values for a configuration MBean. This command cannot be used for Runtime MBeans.

If the command is successful, it returns `OK`.

When you use this command for an Administration MBean, the new values are saved to the `config.xml` file.

We recommend that you do not use this command to set values on a Local Configuration MBean. If you use this command for a Local Configuration MBean, the new values are not saved to the `config.xml` file. Depending on the attribute that you set, the subsystem that uses the MBean might not be able to modify its operation per the new value. In addition, some subsystems require that their Local Configuration MBeans be replicated throughout a domain. If you modify the value for a Local Configuration MBean on one server, the new value will not be replicated throughout the domain and the subsystem might not operate correctly.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   SET {–type mbeanType|–mbean objectName}
   –property property1 property1_value
   [-property property2 property2_value] . . .
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Specifies the listen address and listen port of the Administration Server. Only the Administration Server can access Administration MBeans. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4. Although the `SET` command supports the `-adminurl`, we recommend that you do not use it to set values of Local Configuration MBeans.
`-type mbeanType`	Sets the properties for all MBeans of a specific type. For more information, refer to "Specifying MBean Types" on page 1-56.

Argument	Definition
`-mbean` *`objectName`*	Fully qualified object name of an MBean in the `WebLogicObjectName` format. For example: *"domain*`:Type=`*type*`,Name=`*name"* For more information, refer to the Javadoc for `WebLogicObjectName`.
`-property` *`property`*	The name of the property to be set.

`property _value`	The value to be set.
	■ When the property value is an MBean array, separate each MBean object name by a semicolon and surround the entire property value list with quotes:
	"domain`:Name=`*name*`,Type=`*type*`;`*domain*`:Name=`*name*`,Type=`*type"*
	■ When the property value is a String array, separate each string by a semicolon and surround the entire property value list with quotes:
	"String1`;`*String2*`;. . . "`
	■ When the property value is a String or String array, you can set the value to null by using either of the following:
	`-property` *property-name* `""`
	`-property` *property-name*
	For example, both `-property ListenAddress ""` and `-property ListenAddress` set the listen address to null.
	■ If the property value contains spaces, surround the value with quotes:
	`"-Da=1 -Db=3"`
	For example:
	`SET -type ServerStart -property Arguments "-Da=1 -Db=3"`
	■ When setting the properties for a JDBC Connection Pool, you must pass the arguments in the following format:
	"`user:`*username*`;password:`*password*`;server:`*servername"*

Example

The following example sets to `64` the `StdoutSeverityLevel` property of the local configuration instance of the `ServerMBean` for a server named `MedRecManagedServer`:

```
java weblogic.Admin -url http://ManagedHost:8001
  -username weblogic -password weblogic
  SET -mbean
```

```
  MedRec:Location=MedRecManagedServer,Name=MedRecManagedServer,
  Type=ServerConfig
 -property StdoutSeverityLevel 64
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, the server instance writes a log message similar to the following:

```
<Sep 16, 2002 12:11:27 PM EDT> <Info> <Logging> <000000> <Log
messages of every severity will be displayed in the shell console.>
```

The command prints `Ok` to standard out.

The following example sets to `64` the `StdoutSeverityLevel` property for all administration instances of `ServerMBean` in the current domain:

```
java weblogic.Admin -url http://AdminHost:7001
  -username weblogic -password weblogic
  SET -type Server -property StdoutSeverityLevel 64
```

Using weblogic.Admin Commands to Create Servers

If you prefer to use the command line or a script to add one or more Managed Servers to an existing domain, you can use the `weblogic.Admin` utility. The following example illustrates how to use `weblogic.Admin` to add a server named ManagedMedRecServer to the sample MedRec domain.

The example assumes that you are working on a Windows computer.

1. Start the MedRec domain and Administration Server. For example, you can open a command shell and run the following script:
   `WL_HOME``\samples\server\config\medrec\startMedRecServer`

   Where *`WL_HOME`* is the directory in which you installed WebLogic Server.

2. From the same computer on which you started the Administration Server, open a command shell and enter the following command:
   `WL_HOME``\server\bin\setWLSEnv.cmd`

   The `setWLSEnv.cmd` script sets the environment variables that the `weblogic.Admin` utility requires.

3. To create a server instance, enter the following command:
```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic CREATE -mbean
MedRec:Type=Server,Name=MedRecManagedServer
```
   If the `CREATE` command succeeds, it creates a server instance named MedRecManagedServer that is configured with default values. Then the command returns `OK`.

4. To verify that command succeeded, enter the following command:
```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic GET -pretty -mbean
MedRec:Type=Server,Name=MedRecManagedServer
```
   The command returns a list of MedRecManagedServer attributes, similar to the following truncated list:
```
MBeanName: "MedRec:Name=MedRecManagedServer,Type=Server"
        AcceptBacklog: 50
        AdministrationPort: 0
        AutoKillIfFailed: false
        AutoRestart: true
        COM: MedRecManagedServer
        COMEnabled: false
```

5. To change the value of the non-SSL listen port, enter the following command:
```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic SET -mbean
MedRec:Type=Server,Name=MedRecManagedServer -property
ListenPort 7777
```

6. To verify that command succeeded, enter the following command:
```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic GET -pretty -mbean
MedRec:Type=Server,Name=MedRecManagedServer -property
ListenPort
```
   If the command succeeds, it returns the following:
```
MBeanName: "MedRec:Name=MedRecManagedServer,Type=Server"
        ListenPort: 7777
```
   For more information about the server attributes that you can set, refer to the Javadoc for `weblogic.management.configuration.ServerMBean`.

7. To change the value of the SSL listen port, enter the following command:

```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic SET -mbean
MedRec:Name=MedRecServer,Server=MedRecServer,Type=SSL -property
ListenPort 7778
```

8. To verify that command succeeded, enter the following command:

```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic GET -pretty -mbean
MedRec:Name=MedRecServer,Server=MedRecServer,Type=SSL -property
ListenPort
```

   If the command succeeds, it returns the following:

```
MBeanName:
"MedRec:Name=MedRecServer,Server=MedRecServer,Type=SSL"
        ListenPort: 7312
```

   For more information about the SSL attributes that you can set, refer to the Javadoc for `weblogic.management.configuration.SSLMBean`.

9. To enable the server to be started by the Node Manager, enter the following commands:

```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic CREATE -mbean
MedRec:Type=Machine,Name=MyMachine
```

```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic CREATE -mbean
MedRec:Type=NodeManager,Name=MyMachine
```

```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic SET -mbean
MedRec:Type=Server,Name=MedRecManagedServer -property Machine
MedRec:Name=MyMachine,Type=Machine
```

```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic SET -mbean
MedRec:Name=MedRecManagedServer,Server=MedRecManagedServer,Type
=ServerStart -property Username weblogic
```

```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic SET -mbean
MedRec:Name=MedRecManagedServer,Server=MedRecManagedServer,Type
=ServerStart -property Password weblogic
```

For more information about configuring the arguments that the Node Manager uses to start the server, refer to the Javadoc for `weblogic.management.configuration.ServerStartMBean`.

Running Commands in Batch Mode

By default, each `weblogic.Admin` command that you invoke starts a JVM, acts on a server instance, and then shuts down the JVM. To improve performance for issuing several `weblogic.Admin` commands in an uninterrupted sequence, you can use the `BATCHUPDATE` command to run multiple commands in batch mode. The `BATCHUPDATE` command starts a JVM, runs a list of commands, and then shuts down the JVM.

For example, if a domain contains multiple server instances, you can create a file that returns the listen ports of all Managed Servers in a domain. Then you specify this file as an argument in `weblogic.Admin BATCHUPDATE` command.

BATCHUPDATE

Runs a sequence of `weblogic.Admin` commands. All output from commands that `BATCHUPDATE` runs is printed to standard out.

Using this command provides better performance than issuing a series of individual `weblogic.Admin` commands. For more information, refer to the previous section, "Running Commands in Batch Mode" on page 1-76.

Syntax

```
java weblogic.Admin [ [-url URL] | [-adminurl URL] ]
   -username username -password password
    BATCHUPDATE -batchFile fileLocation
   [-continueOnError] [-batchCmdVerbose]
```

Argument	Definition

`{-url [protocol://]listen-address:listen-port}`
or
`{-adminurl [protocol://]Admin-Server-listen-address:listen-port}`
	If the batch file contains commands that access Administration MBeans, use `-url` to specify the Administration Server's listen address and listen port.
	If the batch file contains commands that access Local Configuration or Runtime MBeans, use one of the following:
	■ `-url` to specify the listen address and listen port of the server instance on which you want to access MBeans.
	■ `-adminurl` to access all Local Configuration or Runtime MBeans in the domain.
	If you specify a secure listen port, you must also specify a secure protocol.
	If you do not specify a value, the command assumes `t3://localhost:7001`.
	For more information, refer to the `-url` and `-adminurl` entries in Table 1-1 on page 5 and "Protocol Support" on page 1-4.

`-batchfile fileLocation`	The name of a text file that contains a list of `weblogic.Admin` commands. If you use a relative pathname, the root context is the directory from which you issue the `weblogic.Admin BATCHUPDATE` command. The file must contain one or more commands, formatted as follows: *`COMMAND-NAME arguments`* Place each command on a separate line. Within the batch file, the `BATCHUPDATE` command ignores any line that begins with a # character.

Argument	Definition
-continueOnError	If one of the commands fails or emits errors, weblogic.Admin ignores the error and continues to the next command. By default, weblogic.Admin stops processing commands as soon as it encounters an error.
-batchCmdVerbose	Causes BATCHUPDATE to indicate which command it is currently invoking. As it invokes a command, BATCHUPDATE prints the following to standard out: Executing command: *command-from-batchfile*

Example

This example uses the BATCHUPDATE command to return the listen ports for a collection of server instances in a domain. A file named commands.txt contains the following lines:

```
get -mbean MedRec:Name=MedRecServer,Type=Server -property ListenPort
get -mbean MedRec:Name=MedRecManagedServer,Type=Server -property ListenPort
```

The following command invokes the commands in commands.txt:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
-password weblogic BATCHUPDATE -batchFile c:\commands.txt
-continueOnError -batchCmdVerbose
```

If the command succeeds it outputs the following to standard out:

```
Executing command: get -mbean MedRec:Name=MedRecServer,Type=Server -property
ListenPort
{MBeanName="MedRec:Name=MedRecServer,Type=Server"{ListenPort=7001}}
Executing command: get -mbean MedRec:Name=MedRecManagedServer,Type=Server
-property ListenPort
{MBeanName="MedRec:Name=MedRecManagedServer,Type=Server"{ListenPort=7021}}
```

For information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

For a sample BATCHUPDATE script that creates a simple cluster, refer to "Using BATCHUPDATE to Create a Simple Cluster" on page 1-88.

Commands for Working with Clusters

Table 1-6 is an overview of the commands for working with clusters. Subsequent sections describe command syntax and arguments, and provide an example for each command.

In addition, the section "Using BATCHUPDATE to Create a Simple Cluster" on page 1-88, provides a sample script that uses the BATCHUPDATE command to create a simple cluster in the MedRec domain.

Table 1-6 MBean Management Command Overview

Command	Description
`CLUSTERSTATE`	Returns the number and state of servers in a cluster. See "CLUSTERSTATE" on page 1-80.
`MIGRATE`	Migrates a JMS service or a JTA service from one server instance to another within a cluster. See "MIGRATE" on page 1-81.
`STARTCLUSTER`	Starts all servers in a cluster See "STARTCLUSTER" on page 1-83.
`STOPCLUSTER`	Stops all servers in a cluster. See "STOPCLUSTER" on page 1-85.
`VALIDATECLUSTERCONFIG`	Parses the domain's configuration file and reports any discrepancies in all cluster-related elements. See "VALIDATECLUSTERCONFIG" on page 1-87.

CLUSTERSTATE

Returns the number and state of servers in a cluster.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   CLUSTERSTATE -clusterName clusterName
```

Argument	Definition
{-url [*protocol*://]*listen-address*:*listen-port*}	Specify the listen address and listen port of any server instance that is currently active and that belongs to the cluster. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes t3://localhost:7001. For more information, refer to the -url entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
-clusterName *clusterName*	The name of the cluster as specified in the domain's configuration file (config.xml).

Example

The following example returns information about a cluster:

```
java weblogic.Admin -url AdminHost:7001
  -username weblogic -password weblogic
  CLUSTERSTATE -clustername MedRecCluster
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it returns output similar to the following:

```
There are 3 server(s) (MedRecServer, MedRecManagedServer,
ManagedServer2) in cluster MedRecCluster

Out of which MedRecServer, MedRecManagedServer, ManagedServer2 are
alive
```

MIGRATE

Migrates a JMS service or a JTA Transaction Recovery service to a targeted server within a server cluster.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
  MIGRATE [-jta]
  -migratabletarget(migratabletargetName|servername)
  -destination servername [-sourcedown] [-destinationdown]
```

Argument	Definition
{-url [*protocol*://]*listen-address*:*listen-port*}	Specify the listen address and listen port of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes t3://localhost:7001. For more information, refer to the -url entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
-jta	Specifies that the migration is a migration of JTA services.
-migratabletarget	Names a configuration file identified with the server from which services will migrate. For each server, WebLogic Server auto-creates a migratable target file named: ■ "(servername)_migratable" for JMS ■ servername for JTA This migratable target file is a configuration file that specifies the preferred servers for JMS service and JTA Transaction Recovery service.
-destination	Names the server to which the services will migrate.
-sourcedown	Specifies that the source server is down. This switch should be used very carefully. If the source server is not in fact down, but only unavailable because of network problems, the service will be activated on the destination server without being removed from the source server, resulting in two simultaneous running versions of the same service, *which could cause corruption of the transaction log or of JMS messages*.

Argument	Definition
-destinationdown	Specifies that the destination server is down. A JMS service migrated to a non-running server will be lost. When migrating the JTA Transaction Recovery Service to a non-running server, the target server will assume recovery services when it is started.

Examples

In the following example, a JMS service is migrated from myserver2 to myserver3.

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic
   MIGRATE -migratabletarget myserver2_migratable
  -destination myserver3
```

In the following example, a JTA Transaction Recovery service is migrated from myserver2 to myserver3.

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic
    MIGRATE -jta -migratabletarget myserver2
   -destination myserver3 -sourcedown
```

STARTCLUSTER

Starts all of the servers that are in a cluster have been configured to use a Node Manager. When the command finishes, all servers in the cluster are in the `RUNNING` state.

This command requires the following environment:

- The domain's Administration Server must be running.
- The Node Manager must be running on the Managed Server's host machine.
- The Managed Server's startup items and Node Manager settings must be set up as described in "Managing Server Availability with Node Manager" in the *Configuring and Managing WebLogic Server* guide.

Note: In the Administration Console, the Servers—General tab includes a Startup Mode field that you use to specify the state in which a server starts. However, this setting only applies if you start a server from the local host using the `weblogic.Server` command. The Node Manager, and therefore the `weblogic.Admin STARTCLUSTER` command, does not use the value that you specify. For example, even if you specify `STANDBY` as the value for the Startup Mode, if you issue the `weblogic.Admin STARTCLUSTER` command, the servers will start in the `RUNNING` state.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   STARTCLUSTER -clusterName clusterName
```

Argument	Definition
`{-url [protocol://]listen-address:listen-port}`	Specify the listen address and listen port of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`-clusterName clusterName`	The name of the cluster as specified in the domain's configuration file (`config.xml`).

Example

The following example starts a cluster:

```
java weblogic.Admin -url AdminHost:7001
  -username weblogic -password weblogic
  STARTCLUSTER -clustername MedRecCluster
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it returns output similar to the following:

```
Starting servers in cluster MedRecCluster: MedRecMS2,MedRecMS1
All servers in the cluster "MedRecCluster" started successfully.
```

STOPCLUSTER

Gracefully shuts down all servers in a cluster.

A graceful shutdown gives WebLogic Server subsystems time to complete certain application processing currently in progress. For information, refer to "Graceful Shutdown" in the *Configuring and Managing WebLogic Server* guide.

If a Node Manager started a server instance, and if the server does not respond to the graceful shutdown request, the Node Manager forcefully shuts down the server.

Syntax

```
java weblogic.Admin [-url URL]
   -username username -password password
   STOPCLUSTER -clusterName clusterName
```

Argument	Definition
`{-url [protocol://]listen-address:listen-port}`	Specify the listen address and listen port of the Administration Server. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`-clusterName clusterName`	The name of the cluster as specified in the domain's configuration file (`config.xml`).

Example

The following example stops a cluster:

```
java weblogic.Admin -url AdminHost:7001
  -username weblogic -password weblogic
  STOPCLUSTER -clustername MedRecCluster
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

If the command succeeds, it returns output similar to the following:

```
Shutting down servers in cluster MedRecCluster: MedRecMS2,MedRecMS1
All servers in the cluster "MedRecCluster" were shutdown
successfully
```

VALIDATECLUSTERCONFIG

Parses the domain's configuration file and reports any errors in the configuration of cluster-related elements.

You can run this command only on a WebLogic Server host that can access the domain's configuration file through the host's file system.

Syntax

```
java weblogic.Admin [-url URL]
    -username username -password password
   VALIDATECLUSTERCONFIG
   -configPath pathname
```

Argument	Definition
`{-url [protocol://]listen-address:listen-port}`	Specify the listen address and listen port of any active server in the domain, regardless of whether it belongs to a cluster. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes `t3://localhost:7001`. For more information, refer to the `-url` entry in Table 1-1 on page 5 and "Protocol Support" on page 1-4.
`-configPath pathname`	The path and file name of the domain's configuration file. A relative pathname is resolved to the directory in which you issue the `VALIDATECLUSTERCONFIG` command.

Example

The following example validates the cluster-related configuration elements for the MedRec domain. In this example, the command is issued from the *WL_HOME* directory:

```
java weblogic.Admin -url AdminHost:7001
  -username weblogic -password weblogic
  VALIDATECLUSTERCONFIG -configpath
  samples\server\config\medrec\config.xml
```

For more information about the environment in which this example runs, refer to "Example Environment" on page 1-7.

Using BATCHUPDATE to Create a Simple Cluster

The `weblogic.Admin BATCHUPDATE` command runs a sequence of `weblogic.Admin` commands that you specify in a text file. This section describes how to use `BATCHUPDATE` to create a simple example cluster. In this example cluster, all server instances run on the same WebLogic Server host as the Administration Server.

Before you can instantiate a cluster, your WebLogic Server license must include a cluster license. If you do not have a cluster license, contact your BEA sales representative. For more information about creating clusters, refer to "Setting Up Clusters" in the *Using WebLogic Server Clusters* guide.

To use `BATCHUPDATE` to create a simple cluster in the MedRec domain, do the following:

1. Start the MedRec domain and Administration Server. For example, you can open a command shell and run the following script:
   `WL_HOME``\samples\server\config\medrec\startMedRecServer` (Windows)
   `WL_HOME``/samples/server/config/medrec/startMedRecServer.sh` (UNIX)

   Where *`WL_HOME`* is the directory in which you installed WebLogic Server.

2. In a command shell, enter the following command:
   `WL_HOME``\server\bin\setWLSEnv.cmd` (Windows)
   `WL_HOME``/server/bin/setWLSEnv.sh` (UNIX)

3. Copy the commands in Listing 1-1 and paste them into an empty text file. Make sure that each command is on a separate, single line. For example, "`SET -mbean MedRec:Type=WebServer,Name=MedRecMS1,Server=MedRecMS1 -property LoggingEnabled true`" must be on a single line.

4. Save the text file.

5. Edit the commands that you pasted into the text file as follows:

   - In the command `CREATE -mbean MedRec:Type=Machine,Name=calamine`, change **`calamine`** to refer either to the name or IP address of the computer that is running the Administration Server.
   - If the listen ports 7777 and 7778 are already in use, change the port numbers in the commands.

- If the IP address `239.0.0.32` is already in use, change the address to a valid multicast address. For information about multicast addresses, refer to "Communications in a Cluster" in the *Using WebLogic Server Clusters* guide.

6. In the command shell, enter the following command:
   ```
   java weblogic.Admin -url localhost:7001 -username weblogic
   -password weblogic BATCHUPDATE -batchFile filename
   -continueonerror -batchCmdVerbose
   ```
   where *`filename`* is the name of the file that you created in step 4.

   Note: The above command assumes that you are running the MedRec server and the `BATCHUPDATE` command on the same Windows computer, and that you have not modified the default listen port of 7001. If you specified some other listen address or listen port for the MedRec Administration Server, use the `-url` argument to specify your modified address and listen port.

   The `BATCHUPDATE` command returns `OK` for each command that it successfully runs.

To verify that you successfully created a cluster, view the Administration Console. In the left pane of the Administration Console, open the Cluster folder and make sure that it contains a cluster named MedRecCluster. Make sure that the cluster contains two server instances named MedRecMS1 and MedRecMS2. Also verify that the servers are targeted for the machine that the command `CREATE -mbean MedRec:Type=Machine,Name=calamine` (from Listing 1-1) creates.

Listing 1-1 BATCHUPDATE Commands for Creating a Cluster

```
#Create Server Instances
CREATE -mbean MedRec:Type=Server,Name=MedRecMS1
CREATE -mbean MedRec:Type=Server,Name=MedRecMS2

#Configure Servers
SET -mbean MedRec:Type=Server,Name=MedRecMS1 -property ListenPort 7777
SET -mbean MedRec:Type=WebServer,Name=MedRecMS1,Server=MedRecMS1 -property
LoggingEnabled true
SET -mbean MedRec:Type=Server,Name=MedRecMS2 -property ListenPort 7778
SET -mbean MedRec:Type=WebServer,Name=MedRecMS2,Server=MedRecMS2 -property
LoggingEnabled true

#Create and Configure Cluster
CREATE -mbean MedRec:Type=Cluster,Name=MedRecCluster
SET -mbean MedRec:Type=Cluster,Name=MedRecCluster -property MulticastAddress
239.0.0.32
```

```
SET -mbean MedRec:Type=Server,Name=MedRecMS1 -property Cluster
MedRec:Name=MedRecCluster,Type=Cluster
SET -mbean MedRec:Type=Server,Name=MedRecMS2 -property Cluster
MedRec:Name=MedRecCluster,Type=Cluster

#Deploy Resources to Cluster
INVOKE -mbean MedRec:Name=MedRecPool,Type=JDBCConnectionPool -method addTarget
MedRec:Name=MedRecCluster,Type=Cluster
INVOKE -mbean MedRec:Name=MedRecTxPool,Type=JDBCConnectionPool -method addTarget
MedRec:Name=MedRecCluster,Type=Cluster
INVOKE -mbean MedRec:Name=MedRecTxDataSource,Type=JDBCTxDataSource -method
addTarget MedRec:Name=MedRecCluster,Type=Cluster
INVOKE -mbean MedRec:Name=MedRecDataSource,Type=JDBCTxDataSource -method
addTarget MedRec:Name=MedRecCluster,Type=Cluster

INVOKE -mbean MedRec:Name=Queue,Type=JMSConnectionFactory -method addTarget
MedRec:Name=MedRecCluster,Type=Cluster
INVOKE -mbean MedRec:Name=Topic,Type=JMSConnectionFactory -method addTarget
MedRec:Name=MedRecCluster,Type=Cluster

INVOKE -mbean MedRec:Name=Topic,Type=JMSConnectionFactory -method addTarget
MedRec:Name=MedRecCluster,Type=Cluster
INVOKE -mbean MedRec:Name=Queue,Type=JMSConnectionFactory -method addTarget
MedRec:Name=MedRecCluster,Type=Cluster

INVOKE -mbean MedRec:Name=MedRecJMSServer,Type=JMSServer -method addTarget
MedRec:Name=MedRecCluster,Type=Cluster

#Configure Machines and Node Manager
CREATE -mbean MedRec:Type=Machine,Name=calamine
CREATE -mbean MedRec:Type=NodeManager,Name=calamine
SET -mbean MedRec:Type=Server,Name=MedRecMS1 -property Machine
MedRec:Name=calamine,Type=Machine
SET -mbean MedRec:Type=Server,Name=MedRecMS2 -property Machine
MedRec:Name=calamine,Type=Machine

CREATE -mbean  MedRec:Name=MedRecMS1,Server=MedRecMS1,Type=ServerStart
SET -mbean MedRec:Name=MedRecMS1,Server=MedRecMS1,Type=ServerStart -property
Username weblogic
SET -mbean MedRec:Name=MedRecMS1,Server=MedRecMS1,Type=ServerStart -property
Password weblogic

CREATE -mbean MedRec:Name=MedRecMS2,Server=MedRecMS1,Type=ServerStart
SET -mbean MedRec:Name=MedRecMS1,Server=MedRecMS1,Type=ServerStart -property
Username weblogic
SET -mbean MedRec:Type=ServerStart,Name=MedRecMS2,Server=MedRecMS2 -property
Password weblogic
```

Deploying Applications and Starting Servers in the Simple Cluster

After you create the cluster, you can use either the Administration Server or the `weblogic.Deployer` utility to deploy applications to the cluster. For more information, refer to "Deployment Tools Reference" in the *Deploying WebLogic Server Applications* guide and "Deploying Applications and Modules" in the Administration Console Online Help.

The commands in Listing 1-1 enable the MedRecMS1 and MedRecMS2 server instances to be started by the Node Manager. For information on setting up Managed Servers to be started by a Node Manager, refer to the following sections in the *Configuring and Managing WebLogic Server* guide:

- "Configure a Machine to Use Node Manager"
- "Configure Managed Server Startup Arguments"
- "Starting Node Manager"

2 Using the WebLogic Server Java Utilities

WebLogic Server provides several Java programs that simplify installation and configuration tasks, provide services, and offer convenient shortcuts. The following sections describe each Java utility provided with WebLogic Server Server. The command-line syntax is specified for all utilities and, for some, examples are provided.

- myip
- NetAddresses
- pem2der
- Schema
- showLicenses
- system
- t3dbping
- verboseToZip
- version
- writeLicense

To use these utilities you must correctly set your `CLASSPATH`. For more information, see "Setting the Classpath."

AppletArchiver

The `AppletArchiver` utility runs an applet in a separate frame, keeps a record of all of the downloaded classes and resources used by the applet, and packages these into either a `.jar` file or a `.cab` file. (The `cabarc` utility is available from Microsoft.)

Syntax

```
$ java utils.applet.archiver.AppletArchiver URL filename
```

Argument	Definition
URL	URL for the applet.
filename	Local filename that is the destination for the `.jar/.cab` archive.

CertGen

The `CertGen` utility generates certificates that should only be used for demonstration or testing purposes and not in a production environment.

Syntax

```
$ java utils.CertGen password certfile keyfile [export]
```

Argument	Definition
password	Defines the password for the private key.
certfile	Defines the directory in which to copy the generated certificate file.
keyfile	Defines the directory in which to copy the generated private key file.
`export`	By default, the `CertGen` utility generates domestic strength certificates. Specify the [export] option if you want the tool to generate export strength certificates.

Example

To generate a certificate:

1. Copy the following files to the directory in which you run the CertGen tool:
   - `WL_HOME/server/lib/CertgenCA.der`—The certificate for a certificate authority trusted by WebLogic Server.
   - `WL_HOME/server/lib/CertGenCAKey.der`—The private key for a certificate authority trusted by WebLogic Server.
2. Enter the following command to generate certificate files named `testcert` with private key files named `testkey`:

```
$ java utils.CertGen mykeypass testcert testkey
Creating Domestic Key Strength - 1024

Encoding
.............................................................
.............................................................
.............................................................
Created Private Key files - testkey.der and testkey.pem
```

```
Encoding
.....................................................................
.....................................................................
.....................................................................
Created Certificate files - testcert.der and testcert.pem
.....................................................................
```

appc

This utility compiles and validates a J2EE EAR file, an EJB JAR file or a WAR file for deployment.

For more information, see WebLogic Server EJB Tools at http://e-docs.bea.com/wls/docs81b/ejb/EJB_tools.html#1087034.

Syntax

```
java weblogic.appc [options] <EAR, JAR, or WAR file or directory>
```

Argument	Definition
O -print	Prints the standard usage message.
-version	Prints jspc version information.
-output <file>	Specifies an alternate output archive or directory. If not set, the output is placed in the source archive or directory.
-forceGeneration	Forces generation of EJB and JSP classes. Without this flag, the classes may not be regenerated (if determined to be unnecessary).
-lineNumbers	Adds line numbers to generated class files to aid in debugging.
-basicClientJar	Does not include deployment descriptors in client JARs generated for EJBs.
-idl	Generates IDL for EJB remote interfaces.
-idlOverwrite	Always overwrites existing IDL files.
-idlVerbose	Displays verbose information for IDL generation.
-idlNoValueTypes	Does not generate valuetypes and the methods/attributes that contain them.
-idlNoAbstractInterfaces	Does not generate abstract interfaces and methods/attributes that contain them.
-idlFactories	Generates factory methods for valuetypes.

Argument	Definition
-idlVisibroker	Generates IDL somewhat compatible with Visibroker 4.5 C++.
-idlOrbix	Generates IDL somewhat compatible with Orbix 2000 2.0 C++.
-idlDirectory <dir>	Specifies the directory where IDL files will be created (default: target directory or JAR)
-idlMethodSignatures <>	Specifies the method signatures used to trigger IDL code generation.
-iiop	Generates CORBA stubs for EJBs.
-iiopDirectory <dir>	Specifies the directory where IIOP stub files will be written (default: target directory or JAR)
-keepgenerated	Keeps the generated .java files.
-compiler <javac>	Selects the Java compiler to use.
-g	Compiles debugging information into a class file.
-O	Compiles with optimization on.
-nowarn	Compiles without warnings.
-verbose	Compiles with verbose output.
-deprecation	Warns about deprecated calls.
-normi	Passes flags through to Symantec's sj.
-J<option>	Passes flags through to Java runtime.
-classpath <path>	Selects the classpath to use during compilation.
-advanced	Prints advanced usage options.

Conversion

If you have used a pre-6.0 version of WebLogic Server, you must convert your `weblogic.properties` files. Instructions for converting your files using a conversion script are available in the Administration Console Online Help section called "Conversion."

der2pem

The `der2pem` utility converts an X509 certificate from DER format to PEM format. The `.pem` file is written in the same directory as the source `.der` file.

Syntax

```
$ java utils.der2pem derFile [headerFile] [footerFile]
```

Argument	Description
derFile	The name of the file to convert. The filename must end with a `.der` extension, and must contain a valid certificate in `.der` format.
headerFile	The header to place in the PEM file. The default header is "-----BEGIN CERTIFICATE-----". Use a header file if the DER file being converted is a private key file, and create the header file containing one of the following: ■ "-----BEGIN RSA PRIVATE KEY-----" for an unencrypted private key. ■ "-----BEGIN ENCRYPTED PRIVATE KEY-----" for an encrypted private key. **Note:** There must be a new line at the end of the header line in the file.
footerFile	The header to place in the PEM file. The default header is "-----END CERTIFICATE-----". Use a footer file if the DER file being converted is a private key file, and create the footer file containing one of the following in the header: ■ "-----END RSA PRIVATE KEY-----" for an unencrypted private key. ■ "-----END ENCRYPTED PRIVATE KEY-----" for an encrypted private key. **Note:** There must be a new line at the end of the header line in the file.

Example

```
$ java utils.der2pem graceland_org.der
Decoding
.....................................................................
```

dbping

The `dbping` command-line utility tests the connection between a DBMS and your client machine via a JDBC driver. You must complete the installation of the driver before attempting to use this utility. For more information on how to install a driver, see WebLogic jDrivers at http://e-docs.bea.com/wls/docs81b/jdrivers.html.

Syntax

```
$ java -Dbea.home=license_location utils.dbping DBMS user password DB
```

Argument	Definition
license_location	The directory containing your WebLogic Server license (`license.bea`). For example, `d:\beaHome\`. Required only if using a BEA-supplied JDBC driver.

DBMS	Choose one of the following for your JDBC driver:
	WebLogic jDriver for Microsoft SQL Server:
	`MSSQLSERVER4`
	WebLogic jDriver for Oracle:
	`ORACLE`
	Oracle Thin Driver:
	`ORACLE_THIN`
	Sybase JConnect driver:
	`JCONNECT`
	Sybase JConnect 5.5 (JDBC 2.0) driver:
	`JCONN2`

user	Valid username for login. Use the same values you use with `isql` or `sqlplus`.
password	Valid password for the user. Use the same values you use with `isql` or `sqlplus`.

Argument	Definition
DB	Name of the database. Use the following format, depending on which JDBC driver you use: WebLogic jDriver for Microsoft SQL Server: *DBNAME*@*HOST*:*PORT* WebLogic jDriver for Oracle: *DBNAME* Oracle Thin Driver: *HOST*:*PORT*:*DBNAME* Sybase JConnect driver: JCONNECT: *HOST*:*PORT*/*DBNAME* Sybase JConnect driver: JCONN2: *HOST*:*PORT*/*DBNAME* Where: ■ *HOST* is the name of the machine hosting the DBMS, ■ *PORT* is port on the database host where the DBMS is listening for connections, and ■ *DBNAME* is the name of a database on the DBMS. (For Oracle, this is the name of a DBMS defined in the tnsnames.ora file.)

Example

```
$ C:\bea\weblogic700b\samples\server\config\examples>java
utils.dbping ORACLE_THIN scott tiger lcdbsol1:1561:lcs901


**** Success!!! ****

You can connect to the database in your app using:

java.util.Properties props = new java.util.Properties();
props.put("user",     "scott");
props.put("password", "tiger");

java.sql.Driver d =
(java.sql.Driver)Class.forName("oracle.jdbc.driver.OracleD
river").newInstance();
java.sql.Connection conn =
d.connect("jdbc:oracle:thin:@lcdbsol1:1561:lcs901",
```

```
props);

// This mode is superior, especially in serverside classes because
// it avoids DriverManager calls are class synchronized, and will
// bottleneck any other JDBC in the server, even already-running
// connections, because all JDBC drivers useDriverManager.println()
// to log info and exceptions, and that call is also class
   synchronized.

// For repeated connecting, a single driver instance can be re-used.

**** or ****

Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
java.sql.Connection conn =
Driver.connect("jdbc:oracle:thin:@lcdbsol1:1561:lcs901", "scott",
"tiger");

**** or ****

java.util.Properties props = new java.util.Properties();
props.put("user",     "scott");
props.put("password", "tiger");
Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
java.sql.Connection conn =
Driver.connect("jdbc:oracle:thin:@lcdbsol1:1561:lcs901", props);
```

DDInit

DDInit is a utility for generating deployment descriptors for applications to be deployed on WebLogic Server. Target a module's archive or folder and DDInit uses information from the module's class files to create appropriate deployment descriptor files.

WebLogic Builder, the graphical user interface for generating and editing deployment descriptors, runs DDInit to generate deployment descriptors. See WebLogic Builder for more information.

In its command-line version, unlike in WebLogic Builder, DDInit writes new files that overwrite existing descriptor files. If META-INF (for EAR or EJB), or WEB-INF (for Web Applications), does not exist, DDInit creates it.

Specify the type of J2EE deployable for which you want deployment descriptors generated by using the DDInit command command specific to the type, as described below.

EJBInit

Target a JAR file or a folder containing files that you intend to archive as a JAR file, and EJBInit will generate the ejb-jar.xml and the weblogic-ejb-jar.xml files for the module.

```
java weblogic.marathon.ddinit.EJBInit <module>
```

EJBInit looks in folders under the target and finds EJBs (bean class, local or remote home, remote or local interface). Matches interfaces with beans, and determines from that match which home belongs to which bean. In the bean itself it looks for CMP filelds. then for relationships between entity beans. From information gathered in this way, EJBInit writes the deployment descriptors.

DDInit supports EJB 2.0. DDInit will provide accurate results for session beans from 1.1, but is not likely to work for EJB 1.1 entity beans.

WebInit

Target a WAR file or a folder containing files that you intend to archive as a WAR file, and WebInit will create web.xml and weblogic.xml files for the module.

```
java weblogic.marathon.ddinit.WebInit <module>
```

EARInit

Generate an `application.xml` and a `weblogic-application.xml` file for an `EAR` using this command. Target an existing `EAR` or a folder containing `JAR` or `WAR` files you intend to archive into an `EAR` file.

```
java weblogic.marathon.ddinit.EARInit <module>
```

In WebLogic Builder, `EARInitlooks` recursively at the entire tree under the targeted module. On the command line, you need to already have descriptors for the modules contained in the `EAR`. `application.xml` will account for the modules. The generated `weblogic-application.xml` will be an empty placeholder.

Example

This output from this example describes building deployment descriptor files for `ejb_st.jar`.

```
D:\dev\smarticket5\smarticket\bin>java
weblogic.marathon.ddinit.EJBInit ejb_st.jar

Found 4 classes that implement the EnterpriseBean interface

Discovered module type for
D:\dev\smarticket5\smarticket\bin\ejb_st.jar

Found EJB components. Initializing descriptors

Creating desc for bean
com.sun.j2ee.blueprints.smarticket.ejb.customer.CustomerEJB

   *** found remote home:
com.sun.j2ee.blueprints.smarticket.ejb.customer.CustomerHome

   *** found remote interface:
com.sun.j2ee.blueprints.smarticket.ejb.customer.Customer

  Setting prim-key-class to 'java.lang.String'

Adding Entity bean 'CustomerEJB'

Creating desc for bean
com.sun.j2ee.blueprints.smarticket.ejb.localeinfo.LocaleInfoEJB

   *** found remote home:
com.sun.j2ee.blueprints.smarticket.ejb.localeinfo.LocaleInfoHome

   *** found remote interface:
com.sun.j2ee.blueprints.smarticket.ejb.localeinfo.LocaleInfo
```

```
LocaleInfoEJB is a Stateless Session bean

Adding Session bean 'LocaleInfoEJB'

Creating desc for bean
com.sun.j2ee.blueprints.smarticket.ejb.movieinfo.MovieInfoEJB

   *** found remote home:
com.sun.j2ee.blueprints.smarticket.ejb.movieinfo.MovieInfoHome

   *** found remote interface:
com.sun.j2ee.blueprints.smarticket.ejb.movieinfo.MovieInfo

MovieInfoEJB is a Stateless Session bean

Adding Session bean 'MovieInfoEJB'

Creating desc for bean
com.sun.j2ee.blueprints.smarticket.ejb.ticketsales.TicketSalesEJB

   *** found remote home:
com.sun.j2ee.blueprints.smarticket.ejb.ticketsales.TicketSalesHom
e

   *** found remote interface:
com.sun.j2ee.blueprints.smarticket.ejb.ticketsales.TicketSales

TicketSalesEJB is a Stateful Session bean

Adding Session bean 'TicketSalesEJB'

Writing descriptors

Building module with newly created descriptors

Finished building module
```

Deployer

`weblogic.Deployer` deploys J2EE applications and components to WebLogic Servers. For additional information, see Deployment Tools and Procedures at http://e-docs.bea.com/wls/docs81b/programming/deploying.html#1094693.

The `weblogic.Deployer` utility is new in WebLogic Server 7.0, and replaces the earlier `weblogic.deploy` utility, which has been deprecated. For more information about the deprecated `weblogic.deploy` utility, see "Deploying Applications" in the WebLogic Server Administration Guide.

Syntax

```
% java weblogic.Deployer [options]
[-activate|-deactivate|-remove|-cancel|-list] [files]
```

Actions (select one of the following)

Action	Description
activate	Deploys or redeploys the application specified by `-name` to the servers specified by `-targets`.
cancel	Attempts to cancel the task identified by `-id`.
deactivate	Deactivates the application on the target servers. Deactivation suspends the deployed components, leaving staged data in place in anticipation of subsequent reactivation. This command only works in the two-phase deployment protocol.
delete_files	Removes files specified in the file list and leaves the application activated. This is valid only for unarchived applications. You must specify target servers.
deploy	A convenient alias for `-activate`.
examples	Displays example usages of the tool.
help	Prints a help message.
list	Lists the status of the task identified by `-id`.

Action	Description
remove	Physically removes the application and any staged data from the target servers. The components are deactivated and the targets are removed from the applications configuration. If you remove the application entirely, the associated MBeans are also deleted from the system configuration. This command only works with the two-phase deployment model.
undeploy	A convenient alias for `-unprepare`.
unprepare	Deactivates and unloads classes for the application identified by `-name` on the target servers, leaving the staged application files in a state where they may be edited or quickly reloaded.
upload	Transfers the specified source file(s) to the administration server. Use this option when you are on a remote system and want to deploy an application that resides on the remote system. The application files are uploaded to the WebLogic Server administration server prior to distribution to named target servers.
version	Prints version information.

Options

Option	Description
adminurl	https://<server>:<port> is the URL of the administration server. Default is http://localhost:7001.
debug	Turns on debug messages in the output log.
external_stage	Sets the `stagingMethod` attribute on the application Mbean when it is created so that the application will not be staged but the value of the staging path will be used when preparing the application.

Option	Description
id	The task identifier `-id` is a unique identifier for the deployment task. You can specify an `-id` with the `-activate`, `-deactivate`, or `-remove` commands, and use it later as an argument to `-cancel` or `-list`. Make sure the `-id` is unique from all other existing deployment tasks. The system generates an `-id` if you do not specify one.
name	The application `-name` specifies the name of the application being deployed. This can be the name of an existing, configured application or the name to use when creating a new configuration.
nostage	Sets the `no-staging` attribute on the ApplicationMBean, indicating that the application does not require staging. The system assumes the application already resides at the location specified by its Path attribute on the target servers.
nowait	Once the action is initiated, the tool prints the task id and exits. This is used to initiate multiple tasks and then monitor them later using the `-list` action.
password	Specifies the password on the command line. If you do not provide a password, you will be prompted for one.
remote	Signals that `weblogic.Deployer` is not running on the same machine as the administration server and that the source path should be passed through unchanged because it represents the path on the remote server.
source	Archive or directory, specifies the location of the file or directory to be deployed. Use this option to set the application Path. The source option should reference the root directory or archive being deployed. If using upload, the source path is relative to the current directory. Otherwise, it is relative to the administration server root directory—the directory where the `config.xml` file resides.

Option	Description
stage	Sets the `stagingMethod` attribute on the application when it is created so that the application will always be staged. This value overrides the `stagingMethod` attribute on any targeted servers.
targets	<server 1>,...<component>@<server N>, displays a comma-separated list of the server and/or cluster names. Each target may be qualified with a J2EE component name. This enables different components of the archive to deployed on different servers. When specified for an application that is already deployed, this list is an addition to the existing targets. If any existing targets are again specified, the application is redeployed on those targets and deployed on the new ones.
timeout	Seconds. Specifies the maximum time in seconds to wait for the completion of the deployment task. When the time expires, `weblogic.Deployer` prints out the current status of the deployment and exits.
user	User name.
verbose	Displays additional progress messages.

Examples

Examples of `weblogic.Deployer` commands:

- Deploying a New Application
- Redeploying an Application
- Redeploying Part of an Application
- Deactivating an Application
- Undeploying an Application
- Canceling a Deployment Task
- Listing All Deployment Tasks

Deploying a New Application

```
java weblogic.Deployer -adminurl http://admin:7001 -name app
-source /myapp/app.ear -targets server1,server2 -activate
```

Redeploying an Application

```
java weblogic.Deployer -adminurl http://admin:7001 -name app
-activate
```

Redeploying Part of an Application

```
java weblogic.Deployer -adminurl http://admin:7001 -name appname
-targets server1,server2 -activate jsps/*.jsp
```

Deactivating an Application

```
java weblogic.Deployer -adminurl http://admin:7001 -name app
-targets server1 -deactivate
```

Undeploying an Application

```
java weblogic.Deployer -adminurl http://admin:7001 -name app
-targets server -remove -id tag
```

Canceling a Deployment Task

```
java weblogic.Deployer -adminurl http://admin:7001 -cancel -id
tag
```

Listing All Deployment Tasks

```
java weblogic.Deployer -adminurl http://admin:7001 -list
```

EJBGen

EJBGen is an Enterprise JavaBeans 2.0 code generator. You can annotate your Bean class file with javadoc tags and then use EJBGen to generate the Remote and Home classes and the deployment descriptor files for an EJB application, reducing to one the number of EJB files you need to edit and maintain.

If you have installed BEA WebLogic 7.0 examples, see *SAMPLES_HOME\server\src\examples\ejb20\ejbgen* for an example application that uses EJBGen.

For complete documentation of this tool, see EJBGen in WebLogic Server EJB Utilities at http://e-docs.bea.com/wls/docs81b/ejb/EJB_utilities.html#1079050.

getProperty

The `getProperty` utility gives you details about your Java setup and your system. It takes no arguments.

Syntax

```
$ java utils.getProperty
```

Example

```
$ java utils.getProperty
-- listing properties --
user.language=en
java.home=c:\java11\bin\..
awt.toolkit=sun.awt.windows.WToolkit
file.encoding.pkg=sun.io
java.version=1.1_Final
file.separator=\
line.separator=
user.region=US
file.encoding=8859_1
java.vendor=Sun Microsystems Inc.
user.timezone=PST
user.name=mary
os.arch=x86
os.name=Windows NT
java.vendor.url=http://www.sun.com/
user.dir=C:\weblogic
java.class.path=c:\weblogic\classes;c:\java\lib\cla...
java.class.version=45.3
os.version=4.0
path.separator=;
user.home=C:\
```

host2ior

The host2ior utility obtains the Interoperable Object Reference (IOR) of a WebLogic Server.

Syntax

```
java utils.host2ior (hostname) (port)
```

ImportPrivateKey

The `ImportPrivateKey` utility is used to load a private key into a private keystore file.

Syntax

```
$ java utils.ImportPrivateKey keystore keystorepass alias keypass
certfile keyfile
```

Argument	Definition
keystore	Defines the name of the keystore file. A new keystore is created if one does not exist.
keystorepass	Defines the password to open the keystore file.
alias	Defines the name that is used to look up certificates and keys in the keystore.
keypass	Defines the password used to unlock the private key file and to protect the private key in the keystore.
certfile	The name of the certificate associated with the private key.
keyfile	The name of the file holding the protected private key.

Example

Use the following steps to:

- Generate a certificate and private key using the `CertGen` utility
- Create a keystore and store a private key using the `ImportPrivateKey` utility

1. Copy the `WL_HOME/server/lib/CertGenCA.der` file and the `WL_HOME/server/lib/CertGenCAkey.der` file to your working directory.
2. Use the `utils.CertGen` utility to generate a certificate and private key. See Using the CertGen Tool at http://e-docs.bea.com/wls/docs70/secmanage/ssl.html#1165276.

```
java utils.CertGen mykeypass testcert testkey
Creating Domestic Key Strength - 1024

Encoding
.............................................................
.............................................................
.............................................................
Created Private Key files - testkey.der and testkey.pem
Encoding
.............................................................
.............................................................
.............................................................
Created Certificate files - testcert.der and testcert.pem
.............................................................
```

3. Convert the certificate from DER format to PEM format.

```
D:\bea2\weblogic700\samples\server\src>java utils.der2pem
CertGenCA.der
Encoding
.............................................................
.............................................................
```

4. Concatenate the certificate and the Certificate Authority (CA).

```
D:\bea2\weblogic700\samples\server\src>cat testcert.pem
CertGenCA.pem >> newcerts.pem
```

5. Create a new keystore named `mykeystore` and load the private key located in the `testkey.pem` file.

```
D:\bea2\weblogic700\samples\server\src>java utils.ImportPrivateKey
mykeystore mypasswd mykey mykeypass newcerts.pem testkey.pem
Keystore file not found, creating it
```

jhtml2jsp

Converts JHTML files to JSP files. Be sure to inspect results carefully, as this utility is intended to begin the conversion process and, given the unpredictability of the JHTML code, will not necessarily produce flawless translations.

Output is a new JSP file named after the original file.

The HTTP servlets that are auto-generated from JSP pages (when they are run in the server) differ from the regular HTTP servlets that are generated from JHTML. JSP servlets extend `weblogic.servlet.jsp.JspBase`, and so do not have access to the methods available to a regular HTTP servlet.

If your JHTML pages may reference these methods to access the servlet 'context' or 'config' objects, you will need to substitute these methods with the reserved words in JSP that represent these implicit objects.

If your JHTML uses variables that have the same name as the reserved words in JSP, the tool will output a warning. You will need to edit your Java code in the generated JSP page to change the variable name to something other than a reserved word.

Syntax

```
java weblogic.utils.jhtml2jsp -d <directory> filename.jhtml
```

or

```
java weblogic.utils.jhtml2jsp filename.jhtml
```

Argument	Definition
-d	Specify the target directory. If target directory isn't specified, output is written to current directory.

logToZip

The `logToZip` utility searches an HTTP server log file in common log format, finds the Java classes loaded into it by the server, and creates an uncompressed `.zip` file that contains those Java classes. It is executed from the document root directory of your HTTP server.

To use this utility, you must have access to the log files created by the HTTP server.

Syntax

```
$ java utils.logToZip logfile codebase zipfile
```

Argument	Definition
logfile	Required. Fully-qualified pathname of the log file.
codebase	Required. Code base for the applet, or `""` if there is no code base. By concatenating the code base with the full package name of the applet, you get the full pathname of the applet (relative to the HTTP document root).
zipfile	Required. Name of the `.zip` file to create. The resulting `.zip` file is created in the directory in which you run the program. The pathname for the specified file can be relative or absolute. In the examples, a relative pathname is given, so the `.zip` file is created in the current directory.

Examples

The following example shows how a `.zip` file is created for an applet that resides in the document root itself, that is, with no code base:

```
$ cd /HTTP/Serv/docs
$ java utils.logToZip /HTTP/Serv/logs/access "" app2.zip
```

The following example shows how a `.zip` file is created for an applet that resides in a subdirectory of the document root:

```
C:\>cd \HTTP\Serv
C:\HTTP\Serv>java utils.logToZip \logs\applets\classes app3.zip
```

MulticastTest

The `MulticastTest` utility helps you debug multicast problems when configuring a WebLogic Cluster. The utility sends out multicast packets and returns information about how effectively multicast is working on your network. Specifically, `MulticastTest` displays the following types of information via standard output:

1. A confirmation and sequence ID for each message sent out by this server.
2. The sequence and sender ID of each message received from any clustered server, including this server.
3. A missed-sequenced warning when a message is received out of sequence.
4. A missed-message warning when an expected message is not received.

To use `MulticastTest`, start one copy of the utility on each node on which you want to test multicast traffic.

Warning: Do NOT run the `MulticastTest` utility by specifying the same multicast address (the `-a` parameter) as that of a currently running WebLogic Cluster. The utility is intended to verify that multicast is functioning properly before starting your clustered WebLogic Servers.

For information about setting up multicast, see the configuration documentation for the operating system/hardware of the WebLogic Server host. For more information about configuring a cluster, see *Using WebLogic Server Clusters*.

Syntax

```
$ java utils.MulticastTest -n name -a address [-p portnumber]
   [-t timeout] [-s send]
```

Argument	Definition
`-n name`	Required. A name that identifies the sender of the sequenced messages. Use a different name for each test process you start.
`-a address`	Required. The multicast address on which: (a) the sequenced messages should be broadcast; and (b) the servers in the clusters are communicating with each other. (The default for any cluster for which a multicast address is not set is 237.0.0.1.)

Argument	Definition
`-p` *`portnumber`*	Optional. The multicast port on which all the servers in the cluster are communicating. (The multicast port is the same as the listen port set for WebLogic Server, which defaults to 7001 if unset.)
`-t` *`timeout`*	Optional. Idle timeout, in seconds, if no multicast messages are received. If unset, the default is 600 seconds (10 minutes). If a timeout is exceeded, a positive confirmation of the timeout is sent to `stdout`.
`-s` *`send`*	Optional. Interval, in seconds, between sends. If unset, the default is 2 seconds. A positive confirmation of each message sent out is sent to `stdout`.

Example

```
$ java utils.MulticastTest -N server100 -A 237.155.155.1
Set up to send and receive on Multicast on Address 237.155.155.1 on
port 7001
Will send a sequenced message under the name server100 every 2
seconds.
Received message 506 from server100
Received message 533 from server200
   I (server100) sent message num 507
Received message 507 from server100
Received message 534 from server200
   I (server100) sent message num 508
Received message 508 from server100
Received message 535 from server200
   I (server100) sent message num 509
Received message 509 from server100
Received message 536 from server200
   I (server100) sent message num 510
Received message 510 from server100
Received message 537 from server200
   I (server100) sent message num 511
Received message 511 from server100
Received message 538 from server200
   I (server100) sent message num 512
Received message 512 from server100
Received message 539 from server200
   I (server100) sent message num 513
Received message 513 from server100
```

myip

The `myip` utility returns the IP address of the host.

Syntax

```
$ java utils.myip
```

Example

```
$ java utils.myip
Host toyboat.toybox.com is assigned IP address: 192.0.0.1
```

NetAddresses

Syntax

Usage: java utils.t2dbtest username password server weblogic.t2.driver weblogic.t2.url #logins #queries tablename

pem2der

The `pem2der` utility converts an X509 certificate from PEM format to DER format. The `.der` file is written in the same directory as the source `.pem` file.

Syntax

```
$ java utils.pem2der pemFile
```

Argument	Description
pemFile	The name of the file to be converted. The filename must end with a `.pem` extension, and it must contain a valid certificate in `.pem` format.

Example

```
$ java utils.pem2der graceland_org.pem
Decoding
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
```

Schema

The `Schema` utility lets you upload SQL statements to a database using the WebLogic JDBC drivers. For additional information about database connections, see *Programming WebLogic JDBC*.

Syntax

```
$ java utils.Schema driverURL driverClass [-u username]
   [-p password] [-verbose] SQLfile
```

Argument	Definition
driverURL	Required. URL for the JDBC driver.
driverClass	Required. Pathname of the JDBC driver class.
-u *username*	Optional. Valid username.
-p *password*	Optional. Valid password for the user.
-verbose	Optional. Prints SQL statements and database messages.
SQLfile	Required. Text file with SQL statements.

Example

The following code shows a `Schema` command line for the `examples.utils` package:

```
D:\bea\weblogic700\samples\server\src>java utils.Schema
"jdbc:pointbase:server://localhost/demo"
"com.pointbase.jdbc.jdbcUniversalDriver" -u "examples"
-p "examples" examples/utils/ddl/demo.ddl

utils.Schema will use these parameters:

url: jdbc:pointbase:server://localhost/demo
driver: com.pointbase.jdbc.jdbcUniversalDriver
dbserver: null
user: examples
password: examples
SQL file: examples/utils/ddl/demo.ddl
```

showLicenses

The `showLicenses` utility displays license information about BEA products installed in this machine.

Syntax

```
$ java -Dbea.home=license_location utils.showLicenses
```

Argument	Description
license_location	The fully qualified name of the directory where the `license.bea` file exists.

Example

```
$ java -Dbea.home=d:\bea utils.showLicense
```

system

The `system` utility displays basic information about your computer's operating environment, including the manufacturer and version of your JDK, your `CLASSPATH`, and details about your operating system.

Syntax

```
$ java utils.system
```

Example

```
$ java utils.system
* * * * * * * java.version * * * * * * *
1.1.6

* * * * * * * java.vendor * * * * * * *
Sun Microsystems Inc.

* * * * * * * java.class.path * * * * * * *
\java\lib\classes.zip;\weblogic\classes;
\weblogic\lib\weblogicaux.jar;\weblogic\license
...

* * * * * * * os.name * * * * * * *
Windows NT

* * * * * * * os.arch * * * * * * *
x86

* * * * * * * os.version * * * * * * *
4.0
```

t3dbping

The `t3dbping` utility tests a WebLogic JDBC connection to a DBMS via any two-tier JDBC driver. You must have access to a WebLogic Server Server and a DBMS to use this utility.

Syntax

```
$ java utils.t3dbping WebLogicURL username password DBMS
driverClass driverURL
```

Argument	Definition
WebLogicURL	Required. URL of the WebLogic Server.
username	Required. Valid username of DBMS user.
password	Required. Valid password of DBMS user.
DBMS	Required. Database name.
driverClass	Required. Full package name of the WebLogic Server two-tier driver.
driverURL	Required. URL of the WebLogic Server two-tier driver.

verboseToZip

When executed from the document root directory of your HTTP server, `verboseToZip` takes the standard output from a Java application run in verbose mode, finds the Java classes referenced, and creates an uncompressed `.zip` file that contains those Java classes.

Syntax

```
$ java utils.verboseToZip inputFile zipFileToCreate
```

Argument	Definition
inputFile	Required. Temporary file that contains the output of the application running in verbose mode.
zipFileToCreate	Required. Name of the `.zip` file to be created. The resulting `.zip` file is be created in the directory in which you run the program.

UNIX Example

```
$ java -verbose myapplication > & classList.tmp
$ java utils.verboseToZip classList.tmp app2.zip
```

NT Example

```
$ java -verbose myapplication > classList.tmp
$ java utils.verboseToZip classList.tmp app3.zip
```

version

The `version` utility displays version information about your installed WebLogic Server via `stdout`.

Syntax

```
$ java weblogic.Admin -url host:port -username username -password
password VERSION
```

Example

```
$ java weblogic.Admin
-url localhost:7001 -username system -password foo VERSION
```

writeLicense

The `writeLicense` utility writes information about all your WebLogic licenses in a file called `writeLicense.txt`, located in the current directory. This file can then be emailed, for example, to WebLogic technical support.

Syntax

```
$ java utils.writeLicense -nowrite -Dweblogic.system.home=path
```

Argument	Definition
`-nowrite`	Required. Sends the output to `stdout` instead of `writeLicense.txt`.
`-Dweblogic.system.home`	Required. Sets WebLogic system home (the root directory of your WebLogic Server installation). This argument is required unless you are running `writeLicense` from your WebLogic system home.

Examples

```
$ java utils.writeLicense -nowrite
```

Example of UNIX Output

```
* * * * * * System properties * * * * * *

* * * * * * * java.version * * * * * * *
1.1.7

* * * * * * * java.vendor * * * * * * *
Sun Microsystems Inc.

* * * * * * * java.class.path * * * * * * *
c:\weblogic\classes;c:\weblogic\lib\weblogicaux.jar;
c:\java117\lib\classes.zip;c:\weblogic\license
...
```

Example of Windows NT Output

```
* * * * * * * os.name * * * * * * *
Windows NT

* * * * * * * os.arch * * * * * * *
x86

* * * * * * * os.version * * * * * * *
4.0

* * * * * * IP * * * * * *
Host myserver is assigned IP address: 192.1.1.0

* * * * * * Location of WebLogic license files * * * * * *
No WebLogicLicense.class found

No license.bea license found in
weblogic.system.home or current directory

Found in the classpath: c:/weblogic/license/license.bea
Last Modified: 06/02/1999 at 12:32:12

* * * * * * Valid license keys * * * * * *
Contents:
Product Name    : WebLogic
 IP Address     : 192.1.1.0-255
 Expiration Date: never
 Units          : unlimited
 key            : b2fcf3a8b8d6839d4a252b1781513b9
...

* * * * * * All license keys * * * * * *
Contents:
Product Name    : WebLogic
 IP Address     : 192.1.1.0-255
 Expiration Date: never
 Units          : unlimited
 key            : b2fcf3a8b8d6839d4a252b1781513b9
...

* * * * * * WebLogic version * * * * * *
WebLogic Build: 4.0.x xx/xx/1999 10:34:35 #xxxxx
```

CHAPTER

3 weblogic.Server Command-Line Reference

The `weblogic.Server` class is the main class for a WebLogic Server instance. You start a server instance by invoking `weblogic.Server` in a Java command. You can invoke the class directly in a command shell or indirectly through scripts or the Node Manager.

This section describes the following:

For information about using scripts to start an instance of WebLogic Server, refer to "Starting Administration Servers" and "Starting Managed Servers From a WebLogic Server Script" in the Administration Console Online Help.

For information about using the Node Manager to start an instance of WebLogic Server, refer to "Managing Server Availability with Node Manager" in the *Configuring and Managing WebLogic Server* guide.

Required Environment and Syntax for weblogic.Server

This section describes the environment that you must set up before you can start a server instance. Then it describes the syntax for invoking `weblogic.Server`.

Environment

To set up your environment for the `weblogic.Server` command:

1. Install and configure the WebLogic Server software, as described in the *WebLogic Server Installation Guide*. See `http://e-docs.bea.com/wls/docs81b/install/index.html`.
2. Add WebLogic Server classes to the `CLASSPATH` environment variable, as described in "Setting the Classpath" on page 3-2.
3. Include a Java Virtual Machine (JVM) in your `PATH` environment variable. You can use any JVM that is listed in the Supported Configurations page at http://e-docs.bea.com/wls/certifications/certifications/index.html.

   If you do not include a JVM in the `PATH` environment variable, you must provide a pathname for the Java executable file that the JVM provides.

Setting the Classpath

The Java Virtual Machine (JVM) uses a setting called *classpath* to locate essential files and directories.

You can use the following script to set the classpath for a WebLogic Server:

`WL_HOME``\server\bin\setWLSEnv.cmd` (on Windows)
`WL_HOME``/server/bin/setWLSEnv.sh` (on UNIX)

Instead of using `setWLSEnv`, you can use an environment variable or the `-classpath` argument in the startup command. Regardless of the method you choose, include the following in the classpath for the JVM that runs instances of WebLogic Server:

- *WL_HOME*`/server/lib/weblogic_sp.jar`

  Depending on which WebLogic Server release, service pack, or patch that you have installed, this file might not exist on your system. Regardless of whether the file currently exists on your system, we recommend that you include *WL_HOME*`/server/lib/weblogic_sp.jar` in your classpath to ensure compatibility with any updates. You must add this file to the classpath before you add `weblogic.jar`.

- *WL_HOME*`/server/lib/weblogic.jar`

- If you use the trial version of PointBase, an all-Java database management system, then include the following files:

  SAMPLES_HOME`/server/eval/pointbase/server/lib/pbserver41ev.jar` and `pbclient41ev.jar`

  where *SAMPLES_HOME* is *WL_HOME*`/samples`.

- If you use WebLogic Enterprise Connectivity, include the following files:

  WL_HOME`/server/lib/wlepool.jar`
  WL_HOME`/server/lib/wleorb.jar`

  where *WL_HOME* is the directory where you installed WebLogic Server.

Syntax

The syntax for invoking `weblogic.Server` is as follows:

`java [`*options*`] weblogic.Server [-help]`

The `java weblogic.Server -help` command returns a list of frequently used options.

Default Behavior

If you have set up the required environment described in "Environment" on page 3-2, when you enter the command `java weblogic.Server` with no options, WebLogic Server does the following:

1. Looks in the current directory for a file named `config.xml`.
2. If `config.xml` exists in the current directory, WebLogic Server does the following:
   a. If only one server instance is defined in `./config.xml`, it starts that server instance.

      For example, if you issue `java weblogic.Server` from `WL_HOME\samples\server\config\medrec`, WebLogic Server starts the MedRec server.

   b. If there are multiple server instances defined in `./config.xml`, WebLogic Server looks for a server configuration named `myserver`. If it finds such a server configuration, it starts the `myserver` instance.

      If it does not find a server named `myserver`, WebLogic Server exits the `weblogic.Server` process and generates an error message.
3. If there is no `config.xml` file in the current directory, WebLogic Server asks if you want to create a domain and server instance. If you answer yes, WebLogic Server does the following:
   a. Creates an server configuration named `myserver`, and persists the configuration in a file named `./config.xml`.

      Any options that you specify are presisted to the `config.xml` file. For example, if you specify `-Dweblogic.ListenPort=8001`, then WebLogic Server saves `8001` in the config.xml file.

      For any options that you do not specify, the server instance uses default values.

      WebLogic Server uses the username and password that you supply to create a user with administrative privileges.

Note that the server starts as an Administration Server in a new domain. There are no other servers in this domain, nor are any of your deployments or third-party solutions included. You can add them as you would add them to any WebLogic domain.

b. Creates two scripts, `startmydomain.cmd` and `startmydomain.sh`, that you can use to start subsequent instantiations of the server. You can use a text editor to modify startup options such as whether the server starts in production mode or development mode. The `startmydomain` script contains comments that describe each option.

weblogic.Server Configuration Options

You can use `weblogic.Server` options to configure the following attributes of a server instance:

- "JVM Parameters" on page 3-6
- "Location of License and Configuration Data" on page 3-7
- "Server Communication" on page 3-9
- "SSL" on page 3-12
- "Security" on page 3-15
- "Message Output and Logging" on page 3-17
- "Other Server Configuration Options" on page 3-19
- "Clusters" on page 3-21

Unless you are creating a new domain as described in "Using the weblogic.Server Command Line to Create a Domain" on page 3-23, all startup options apply to the current server instantiation; they do not modify the persisted values in an existing `config.xml` file. Use the Administration Console or the `weblogic.Admin` command to modify the `config.xml` file.

For information on verifying the WebLogic Server attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

JVM Parameters

The following table describes frequently used options that configure the Java Virtual Machine (JVM) in which the server instance runs. For a complete list of JVM options, refer to the documentation for your specific JVM. For a list of JVMs that can be used with WebLogic Server, refer to the Supported Configurations page at http://e-docs.bea.com/wls/certifications/certifications/index.html.

Table 3-1 Frequently Used Options for Setting JVM Parameters

Option	Description
`-Xms` and `-Xmx`	Specify the minimum and maximum values (in megabytes) for Java heap memory. For example, you might want to start the server with a default allocation of 200 megabytes of Java heap memory to the WebLogic Server. To do so, you can start the server with the `java -Xms200m` and `-Xmx200m` options. For best performance it is recommended that the minimum and maximum values be the same so that the JVM does not resize the heap. The values assigned to these parameters can dramatically affect the performance of your WebLogic Server and are provided here only as general defaults. In a production environment you should carefully consider the correct memory heap size to use for your applications and environment.
`-classpath`	The minimum content for this option is described under "Setting the Classpath" on page 3-2. Instead of using this argument, you can use an environment variable named `CLASSPATH` to specify the classpath.
`-client` `-server`	Used by some JVMs to start a HotSpot virtual machine, which enhances performance. For a list of JVMs that can be used with WebLogic Server, refer to the Supported Configurations page at http://e-docs.bea.com/wls/certifications/certifications/index.html.

Location of License and Configuration Data

All server instances must have access to license and configuration data. The following table provides options for indicating the location of this data.

Table 3-2 Options for Indicating the Location of License and Configuration Data

Option	Description
`-Dbea.home=`*`bea_home`*	Specifies the location of the BEA home directory, which contains licensing and other essential information. By default, `weblogic.Server` determines the location of the BEA home directory based on values in the classpath.
`-Dweblogic.RootDirectory=`*`path`*	Specifies the server's root directory. By default, the root directory is the directory from which you issue the start command. For more information, refer to "A Server's Root Directory" in the *Configuring and Managing WebLogic Server* guide.

`-Dweblogic.ConfigFile=`*`file_name`*	Specifies a configuration file for your domain. The *`file_name`* value must refer to a valid XML file that conforms to the `config.dtd`. The XML file must exist in the Administration Server's root directory, which is either the current directory or the directory that you specify with `-Dweblogic.RootDirectory`.
	The *`file_name`* value cannot contain a pathname component. For example, the following value is invalid:
	`-Dweblogic.ConfigFile=c:\mydir\myfile.xml`
	Instead, use the following arguments:
	`-Dweblogic.RootDirectory=c:\mydir`
	`-Dweblogic.ConfigFile=myfile.xml`
	For information about `config.dtd`, refer to BEA WebLogic Server Configuration Reference.
	If you do not specify this value, the default is `config.xml` in the server's root directory.

`-Dweblogic.Domain=`*`domain`*	Specifies the name of the domain. This option is not needed unless you are using `weblogic.Server` to create a domain and you want to give the domain a specific name.

For information on how a Managed Server retrieves its configuration data, refer to the `-Dweblogic.management.server` entry in Table 3-3 on page 9.

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

Examples

The following example starts an Administration Server instance named SimpleServer. In the example, the `config.xml` file has been renamed to `SimpleDomain.xml` and it is located in a directory named `c:\my_domains\SimpleDomain`. The command itself is issued from the `D:\` directory after running *`WL_HOME`*`\server\bin\setWLSEnv.cmd`:

```
D:\> java -Dweblogic.Name=SimpleServer
-Dweblogic.ConfigFile=SimpleDomain.xml
-Dweblogic.RootDirectory=c:\my_domains\SimpleDomain
weblogic.Server
```

The following example starts a Managed Server instance named SimpleManagedServer. Specifying a `config.xml` file is not valid because Managed Servers contact the Administration Server for their configuration data. Multiple instances of WebLogic Server can use the same root directory. However, if your server instances share a root directory, make sure that all relative filenames are unique. In this example, SimpleManagedServer shares its root directory with SimpleServer. The command itself is issued from the `D:\` directory after running *`WL_HOME`*`\server\bin\setWLSEnv.cmd`:

```
D:\> java -Dweblogic.Name=SimpleManagedServer
-Dweblogic.management.server=localhost:7001
-Dweblogic.RootDirectory=c:\my_domains\SimpleDomain
weblogic.Server
```

Server Communication

The following table describes the options for configuring how servers communicate.

Table 3-3 Options for Configuring Server Communication

Option	Description
`-Dweblogic.management.server=[protocol]Admin-host:port`	Starts a server instance as a Managed Server and specifies the Administration Server that will configure and manage the server instance. The domain's configuration file does not specify whether a server configuration is an Administration Server or a Managed Server. You determine whether a server instance is in the role of Administration Server or Managed Server with the options that you use to start the instance. If you omit the `-Dweblogic.management.server` option in the start command, the server starts as an Administration Server (although within a given domain, there can be only one active Administration Server instance). Once an Administration Server is running, you must start all other server configurations as Managed Servers by including the `-Dweblogic.management.server` option in the start command. For *`protocol`*, specify HTTP, HTTPS, T3, or T3S. The T3S and HTTPS protocols require you to enable SSL on the Managed Server and the Administration Server and specify the Administration Server's SSL listen port. **Note:** Regardless of which protocol you specify, the initial download of a Managed Server's configuration is over HTTP or HTTPS. After the RMI subsystem initializes, the server instance can use the T3 or T3S protocol. For *`Admin-host`*, specify `localhost` or the DNS name or IP address of the machine where the Administration Server is running. For *`port`*, specify the Administration Server's listen port. If you set up the domain-side administration port, *`port`* must specify the domain-wide administration port. For more information on configuring a connection to the Administration Server, refer to "Configuring a Connection to the Administration Server" in the Administration Console Online Help.

Table 3-3 Options for Configuring Server Communication

Option	Description
`-Dweblogic.ListenAddress=`*`host`*	Specifies the address at which this server instance listens for requests. The *`host`* value must be either the DNS name or the IP address of the computer that is hosting the server instance. This startup option overrides any listen address value specified in the `config.xml` file. The override applies to the current server instantiation; it does not modify the value in the `config.xml` file. Use the Administration Console or the `weblogic.Admin` command to modify the `config.xml` file. We recommend that you specify a known IP address or DNS name and that you use the Administration Console instead of this argument to do so. For more information, refer to "Setting the Listen Address" in the *Configuring and Managing WebLogic Server* guide.
`-Dweblogic.ListenPort=`*`portnumber`*	Enables and specifies the plain-text (non-SSL) listen port for the server instance. This startup option overrides any listen port value specified in the `config.xml` file. The override applies to the current server instantiation; it does not modify the value in the `config.xml` file. Use the Administration Console or the `weblogic.Admin` command to modify the `config.xml` file. The default listen port is 7001. For more information, refer to "Setting the Listen Ports" in the *Configuring and Managing WebLogic Server* guide.
`-Dweblogic.ssl.ListenPort=`*`portnumber`*	Enables and specifies the port at which this WebLogic Server instance listens for SSL connection requests. This startup option overrides any SSL listen port value specified in the `config.xml` file. The override applies to the current server instantiation; it does not modify the value in the `config.xml` file. Use the Administration Console or the `weblogic.Admin` command to modify the `config.xml` file. The default SSL listen port is 7002. For more information, refer to "Setting the Listen Ports" in the *Configuring and Managing WebLogic Server* guide.

Table 3-3 Options for Configuring Server Communication

Option	Description
`-Dweblogic.management.discover={true | false}`	Determines whether an Administration Server recovers control of a domain after the server fails and is restarted. A `true` value causes an Administration Server to refer to its `running-managed-servers.xml` file, which contains information about the deployment state of deployable modules and a list of all Managed Servers that are currently running. When the Administration Server starts with this specified as `true`, it communicates with the Managed Servers and informs them that the Administration Server is running. A `false` value prevents an Administration Server from referring to this file and thus prevents it from communicating with any Managed Servers that are currently active in the domain. **Caution:** Specify `false` for this option only in the development environment of a single server. Specifying `false` can cause server instances in the domain to have an inconsistent set of deployed modules. If you start server instances in Development Mode, the default value is `false`. If you start server instances in Production Mode, the default value is `true`. For more information, refer to "Starting in Development Mode or Production Mode" in the Administration Console Online Help. For information on re-establishing administrative control over Managed Servers after an Administration Server has already started, refer to "DISCOVERMANAGEDSERVER" on page 1-11.

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

SSL

Each Weblogic Server instance uses an instance of `weblogic.management.security.SSLMBean` to represent its SSL configuration. All of the options in the following table that start with `-Dweblogic.security.SSL` modify the configuration of the server's `SSLMBean`. For example, the `-Dweblogic.security.SSL.ignoreHostnameVerification` option sets the value of the `SSLMBean`'s `ignoreHostnameVerification` attribute.

The following table describes the options for configuring a server to communicate using Secure Sockets Layer (SSL).

Table 3-4 Options for Configuring SSL

Option	Description
`-Dweblogic.security.SSL.ignoreHostnameVerification=true`	Disables host-name verification, which enables you to use the demonstration digital certificates that are shipped with WebLogic Server. By default, when a WebLogic Server instance is in the role of SSL client (it is trying to connect to some other server or application via SSL), it verifies that the host name that the SSL server returns in its digital certificate matches the host name of the URL used to connect to the SSL server. If the host names do not match, the connection is dropped. If you disable host name verification, either by using this option or by modifying the server's configuration in the `config.xml` file, the server instance does not verify host names when it is in the role of SSL client. **Note:** BEA does not recommend using the demonstration digital certificates or turning off host name verification in a production environment. This startup option overrides any Host Name Verification setting in the `config.xml` file. The override applies to the current server instantiation; it does not modify the value in the `config.xml` file. Use the Administration Console or the `weblogic.Admin` command to modify the `config.xml` file. For more information, refer to "Using a Hostname Verifier" in the *Managing WebLogic Security* guide.

Table 3-4 Options for Configuring SSL

Option	Description
`-Dweblogic.security.SSL.HostnameVerifier=`*`hostnameverifierimplmentation`*	Specifies the name of a custom Host Name Verifier class. The class must implement the `weblogic.security.SSL.HostnameVerifier` interface.
`-Dweblogic.security.SSL.sessionCache.size=`*`sessionCacheSize`* `-Dweblogic.security.SSL.sessionCache.ttl=`*`sessionCacheTimeToLive`*	Modifies the default server-session caching size and time-to-live for SSL session caching. The *`sessionCacheSize`* value specifies the number of items in session cache and the *`sessionCacheTimeToLive`* value specifies (in seconds) the session cache time-to-live. For `sessionCache.size`: ■ The minimum value is `1` ■ The maximum value is `65537` ■ The default value is `211` For `sessionCache.ttl`: ■ The minimum value is `1` ■ The maximum value is `Integer.MAX_VALUE` ■ The default value is `600`
`-Dweblogic.management.pkpassword=`*`pkpassword`*	Specifies the password for retrieving SSL private keys from an encrypted flat file. Use this option if you store private keys in an encrypted flat file.
`-Dweblogic.security.SSL.trustedCAKeyStore=`*`path`*	Deprecated and ignored by default. If you configure a server instance to use the SSL features that were available before WebLogic Server 8.1, you can use this argument to specify the certificate authorities that the server or client trusts. The *path* value must be a relative or qualified name to the Sun JKS keystore file (contains a repository of keys and certificates). If a server instance is using the SSL features that were available before 8.1, and if you do not specify this argument, the WebLogic Server or client trusts all of the certificates that are specified in *`JAVA_HOME`*`\jre\lib\security\cacerts`. We recommend that you do not use the demonstration certificate authorities in any type of production deployment. For more information, refer to "Configuring the SSL Protocol" in the *Managing WebLogic Security* guide.

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

Setting Additional SSL Attributes

To set additional SSL attributes from the startup command, do the following:

1. To determine which SSL attributes can be configured from startup options, view the WebLogic Server Javadoc for the `SSLMBean` and `ServerMBean`. The Javadoc also indicates valid values for each attribute.

   Each attribute that `SSLMBean` and `ServerMBean` expose as a setter method can be set by a startup option.

2. To set attributes in the `SSLMBean`, add the following option to the start command:
   `-Dweblogic.ssl.`*`attribute-name=value`*

   where *`attribute-name`* is the name of the MBean's setter method without the `set` prefix.

3. To set attributes in the `ServerMBean`, add the following option to the start command:
   `-Dweblogic.server.`*`attribute-name=value`*

   where *`attribute-name`* is the name of the MBean's setter method without the `set` prefix.

For example, the `SSLMBean` exposes its `Enabled` attribute with the following setter method:

```
setEnabled()
```

To enable SSL for a server instance named MedRecServer, use the following command when you start MedRecServer:

```
java -Dweblogic.Name=MedRecServer
     -Dweblogic.ssl.Enabled=true weblogic.Server
```

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

Security

The following table describes the options for configuring general security parameters. Table 3-4 on page 3-12 describes options for setting SSL parameters.

Table 3-5 Options for General Security Parameters

Option	Description
`-Dweblogic.management.username=`*`username`*	Specifies the username under which the server instance will run. The username must belong to a role that has permission to start a server. For information on roles and permissions, refer to "Protecting System Administration Operations" in the *Configuring and Managing WebLogic Server* guide. Instead of using this argument, you can use a boot identity file. For more information, refer to "Bypassing the Prompt for Username and Password" in the Administration Console Online Help.
`-Dweblogic.management.password=`*`password`*	Specifies the user password. Instead of using this argument, you can use a boot identity file. For more information, refer to "Bypassing the Prompt for Username and Password" in the Administration Console Online Help.
`-Dweblogic.system.StoreBootIdentity=true`	Creates a `boot.properties` file in the server's root directory. The file contains the username and an encrypted version of the password that you used to start the server. Do not specify this argument in a server's `ServerStartMBean` (Remote Startup tab in the Administration Console). For more information, refer to "Specifying User Credentials When Starting a Server with the Node Manager" in the Administration Console Online Help. Also, we recommend that you do not add this argument to a startup script. Instead, use it only when you want to create a `boot.properties` file. For more information, refer to "Bypassing the Prompt for Username and Password" in the Administration Console Online Help.

Table 3-5 Options for General Security Parameters

Option	Description
`-Dweblogic.system.BootIdentityFile=`*`filename`*	Specifies a boot identity file that contains a username and password. The *`filename`* value must be the fully qualified pathname of a valid boot identity file. For example: `-Dweblogic.system.BootIdentityFile=C:\BEA\wlserver8.1\user_config\mydomain\myidentity.prop` If you do not specify a filename, a server uses the `boot.properties` in the server's root directory. If there is no boot identity file, the server prompts you to enter a username and password.
`-Dweblogic.system.RemoveBootIdentity=true`	Removes the boot identity file after a server starts.
`-Dweblogic.security.anonymousUserName=`*`name`*	Assigns a user ID to anonymous users. By default, all anonymous users are identified with the string `<anonymous>`. To emulate the security behavior of WebLogic Server 6.x, specify `guest` for the *`name`* value and create a user named `guest` in your security realm. For more information, refer to Defining Users in the *Managing WebLogic Security* guide.
`-Djava.security.manager` `-Djava.security.policy=`*`filename`*	Standard J2EE options that enable the Java 2 security manager and specifiy a filename (using a relative or fully-qualified pathname) that contains Java 2 security policies. To use the WebLogic Server sample policy file, specify *`WL_HOME`*`\server\lib\weblogic.policy`. For more information, refer to Modifying the weblogic.policy File for General Use in the *Managing WebLogic Security* guide.

Table 3-5 Options for General Security Parameters

Option	Description
`-Dweblogic.security.fullyDelegateAuthorization=true`	By default, roles and security policies cannot be set for an EJB or Web application through the Administration Console unless security constraints were defined in the deployment descriptor for the EJB or Web application. Use this command-line argument when starting WebLogic Server to override this problem. This command-line argument does not work with EJBs or EJB methods that use `<unchecked>` or `<restricted>` tags or Web applications that do not have a role-name specified in the `<auth-constraint>` tag.

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

Message Output and Logging

The following table describes options for configuring a server instance's message output.

Table 3-6 Options for Configuring Message Output

Option	Description
`-Dweblogic.Stdout="`*`filename`*`"`	Redirects the JVM's standard output stream to a file. You can specify a pathname that is fully qualified or relative to the WebLogic Server root directory. Use this option to keep a record of the messages from the JVM that are not sent to a WebLogic Server log. For example, a JVM can print `verbosegc` messages to standard out but not to the WebLogic Server log. For more information, refer to "Redirecting JVM Messages to a File" in the Administration Console Online Help.

Table 3-6 Options for Configuring Message Output

Option	Description
`-Dweblogic.Stderr="filename"`	Redirects the JVM's standard error stream to a file. You can specify a pathname that is fully qualified or relative to the WebLogic Server root directory. Use this option to keep a record of the error messages from the JVM that are not sent to a WebLogic Server log. For more information, refer to "Redirecting JVM Messages to a File" in the Administration Console Online Help.

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

Setting Logging Attributes

Each Weblogic Server instance uses an instance of `weblogic.management.configuration.LogMBean` to represent the configuration of its logging services.

To set values for `LogMBean` attributes from the startup command, do the following:

1. To determine which log attributes can be configured from startup options, view the WebLogic Server Javadoc for the `LogMBean`. The Javadoc also indicates valid values for each attribute.

   Each attribute that the `LogMBean` exposes as a setter method can be set by a startup option.

2. Add the following option to the start command:
   `-Dweblogic.log.attribute-name=value`

   where *`attribute-name`* is the name of the MBean's setter method without the `set` prefix.

The `LogMBean` exposes its `FileName` attribute with the following setter method:

```
setFileName()
```

To specify the name of the MedRecServer instance's local log file, use the following command when you start MedRecServer:

```
java -Dweblogic.Name=MedRecServer
     -Dweblogic.log.FileName="C:\logfiles\myServer.log"
     weblogic.Server
```

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

Other Server Configuration Options

The following table describes options for configuring additional attributes of a server instance.

Table 3-7 Options for Configuring Server Attributes

Option	Description
`-Dweblogic.Name=`*`servername`*	Specifies the name of the server instance that you want to start. The specified value must refer to the name of a server that has been defined in the domain's `config.xml` file.
`-Dweblogic.ProductionModeEnabled=` `{true | false}`	Determines whether a server starts in production mode. A `true` value prevents a WebLogic Server from automatically deploying and updating applications that are in the *`domain_name`*`/applications` directory. If you do not specify this option, the assumed value is `false`. For more information, refer to "Starting in Development Mode or Production Mode" in the Administration Console Online Help.

Table 3-7 Options for Configuring Server Attributes

Option	Description
`-Dweblogic.management.startupMode=STANDBY`	Starts a server and places it in the `STANDBY` state. To use this startup argument, the domain must be configured to use the domain-wide administration port. For information about administration ports, refer to "Enabling the Domain-Wide Administration Port" in the *Configuring and Managing WebLogic Server* guide. This startup option overrides any startup mode setting in the `config.xml` file. The override applies to the current server instantiation; it does not modify the value in the `config.xml` file. Use the Administration Console or the `weblogic.Admin` command to modify the `config.xml` file. If you do not specify this value (either on the command line or in `config.xml`), the default is to start in the `RUNNING` state.

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

Setting Additional Server Attributes

Each Weblogic Server instance uses an instance of `weblogic.management.security.ServerMBean` to represent its overall configuration.

To set values for `ServerMBean` attributes from the startup command, do the following:

1. To determine which log attributes can be configured from startup options, view the WebLogic Server Javadoc for the `ServerMBean`. The Javadoc also indicates valid values for each attribute.

   Each attribute that the `ServerMBean` exposes as a setter method can be set by a startup option.

2. Add the following option to the start command:
   `-Dweblogic.server.`*`attribute-name`*`=`*`value`*

   where *`attribute-name`* is the name of the MBean's setter method without the `set` prefix.

The `ServerMBean` exposes its `StdoutSeverityLevel` attribute with the following inherited setter method:

```
setStdoutSeverityLevel()
```

To specify the severity level of messages that the MedRecServer instance prints to standard out, use the following command when you start MedRecServer:

```
java -Dweblogic.Name=MedRecServer
     -Dweblogic.StdoutSeverityLevel=64
     weblogic.Server
```

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

Clusters

Each server in a cluster uses a local instance of `weblogic.management.configuration.ClusterMBean` to represent its view of the cluster configuration. If you want server instances to use a cluster configuration that is different from the values in the `config.xml` file (or that is unspecified in `config.xml`), you can set values of the `ClusterMBean` from the start command.

For example, when you create a cluster, instead of using the Administration Console to define the multicast address, you can leave this attribute undefined. Then, when you start each server instance for the cluster, you use the `-Dweblogic.cluster.MulticastAddress` startup option to specify the multicast address. Because all servers in a cluster must use the same multicast address, you must use the same startup option and value for each server instance in the cluster.

To set a server instance's view of a cluster from the startup command, do the following:

1. To determine which cluster attributes can be configured from startup options, view the WebLogic Server Javadoc for the `ClusterMBean`. The Javadoc also indicates valid values for each attribute.

   Each attribute that the `ClusterMBean` exposes as a setter method can be set by a startup option.

2. Add the following option to the start command:
   `-Dweblogic.cluster.`*`attribute-name`*`=`*`value`*

   where *`attribute-name`* is the name of the MBean's setter method without the `set` prefix.

For example, the `ClusterMBean` exposes its multicast address attribute with the following setter method:

```
setMulticastAddress()
```

To set the multicast address value for a cluster member named MRServer 1, use the following command when you start MRServer1:

```
java -Dweblogic.Name=MRServer1
    -Dweblogic.cluster.MulticastAddress=239.0.0.32 weblogic.Server
```

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, refer to "Verifying Attribute Values That Are Set on the Command Line" on page 3-25.

Using the weblogic.Server Command Line to Start a Server Instance

A simple way to start a server instance is as follows:

1. In a command shell, set up the required environment variables by running the following script:
   `WL_HOME``\server\bin\setWLSEnv.cmd` (on Windows)
   `WL_HOME``/server/bin/setWLSEnv.sh` (on UNIX)

   where *`WL_HOME`* is the directory in which you installed the WebLogic Server software.

2. In the command shell, change to the directory that contains your domain's `config.xml` file. For example, change to the *`WL_HOME`*`\samples\server\config\medrec` directory.

3. To start an Administration Server, enter the following command:
   `java -Dweblogic.Name=`*`servername`* `weblogic.Server`

where *servername* is the name of a server configuration that already exists in the `config.xml` file.

For example, enter the following command to start the MedRec server:

```
java -Dweblogic.Name=MedRecServer weblogic.Server
```

4. If the domain's Administration Server is already running, and if you have already defined a Managed Server in the `config.xml` file, you can start a Managed Server as follows:

   ```
   java -Dweblogic.Name=managed-server-name
   -Dweblogic.management.server=url-for-Administration-Server
   weblogic.Server
   ```

   For example, if you create a Managed Server named MedRecManagedServer in the MedRec domain, you can enter the following command:

   ```
   java -Dweblogic.Name=MedRecManagedServer
   -Dweblogic.management.server=localhost:7001
   weblogic.Server
   ```

Using the weblogic.Server Command Line to Create a Domain

You can use `weblogic.Server` to create a domain that contains a single server instance. You cannot use `weblogic.Server` to add Managed Server instances to a domain, nor can you use `weblogic.Server` to modify an existing domain.

As described in "Default Behavior" on page 3-4, if `weblogic.Server` is unable to find a `config.xml` file, it offers to create the file. Any command option that you specify and that corresponds to an attribute that is persisted in the `config.xml` file will be persisted. For example, the `-Dweblogic.Name` and `-Dweblogic.Domain` options specify the name of a server configuration and the name of a domain. If `weblogic.Server` is unable to find a `config.xml` file, both of these values are persisted in `config.xml`. However, the `-Dweblogic.system.BootIdentityFile` option, which specifies a file that contains user credentials for starting a server instance, is not an attribute that the `config.xml` file persists.

To create and instantiate a simple example domain and server, do the following:

1. In a command shell, set up the required environment variables by running the following script:
   WL_HOME`\server\bin\setWLSEnv.cmd` (on Windows)
   WL_HOME`/server/bin/setWLSEnv.sh` (on UNIX)

   where *WL_HOME* is the directory in which you installed the WebLogic Server software.
2. In the command shell, create an empty directory.
3. In the empty directory, enter the following command:

```
java -Dweblogic.Domain=SimpleDomain -Dweblogic.Name=SimpleServer
-Dweblogic.management.username=weblogic -Dweblogic.management.password=weblogic
-Dweblogic.ListenPort=7701 weblogic.Server
```

After you enter this command, WebLogic Server asks if you want to create a new `config.xml` file. If you enter `y`, it asks you to confirm the password. Then it instantiates a domain named SimpleDomain. The domain's Administration Server is configured as follows:

- The name of the Administration Server is SimpleServer.
- The domain's security realm defines one administrative user, `weblogic`, with a password of `weblogic`.
- For the listen address of the Administration Server, you can use `localhost`, the IP address of the host computer, or the DNS name of the host computer.
- The Administration Server listens on port 7701.

Enter the `weblogic.Server` command as described in this section creates the following files:

- `config.xml`
- `boot.properties` file, which contains the username and password in an encrypted format. This file enables you to bypass the prompt for username and password when you start the server. For more information, refer to "Bypassing the Prompt for Username and Password" in the Administration Console Online Help.
- `startmydomain.cmd` and `startmydomain.sh`, that you can use to start subsequent instantiations of the server.

Verifying Attribute Values That Are Set on the Command Line

To verify that the server instance is using the values that you passed on the command line, use the `weblogic.Admin` utility as follows:

```
java weblogic.Admin -url url-for-server-instance -username username
-password password GET -type MBean-nameConfig -property
attribute-name
```

For example, to determine the multicast address that a cluster member is using, enter the following command, where `MRMachine1:7041` is the listen address and port of the cluster member:

```
java weblogic.Admin -url MRMachine1:7041 -username weblogic
-password weblogic GET -type ClusterConfig -property
MulticastAddress
```

To determine the severity level of messages that the example MedRecServer prints to standard out, enter the following command:

```
java weblogic.Admin -url localhost:7001 -username weblogic
-password weblogic GET -type ServerConfig -property
StdoutSeverityLevel
```

The Administration Console does not display values that you set on the command line because the startup options set attribute values for the server's Local Configuration MBean. For more information about Local Configuration MBeans, refer to "Overview of WebLogic JMX Services" in the *Programming WebLogic Server Management Services with JMX* guide.

For more information on using the weblogic.Admin utility, refer to Chapter 1, "weblogic.Admin Command-Line Reference."

Index

I

J

L

M

P

R

S

T

U

V

W