BEA WebLogic Server®

WebLogic Server Command Reference

Version 9.1
Revised: Dec 15, 2005

Copyright

Restricted Rights Legend

Trademarks and Service Marks

Contents

1. Introduction and Roadmap

Document Scope and Audience 1-1

Guide to This Document 1-1

Related Documentation 1-2

2. weblogic.Admin Command-Line Reference (Deprecated)

Required Environment for the weblogic.Admin Utility 2-2

Syntax for Invoking the weblogic.Admin Utility 2-3

SSL Arguments 2-4

Using SSL to Secure Administration Requests: Main Steps 2-5

Specifying Trust for weblogic.Admin 2-6

Specifying Host Name Verification for weblogic.Admin 2-7

Connection Arguments 2-7

User Credentials Arguments 2-10

Specifying User Credentials 2-12

Examples of Providing User Credentials 2-13

Protocol Support 2-13

Example Environment 2-14

Exit Codes Returned by weblogic.Admin 2-14

Command for Storing User Credentials 2-15

STOREUSERCONFIG 2-15

Syntax 2-15

Configuring the Default Path Name 2-17
Creating User-Configuration and Key Files 2-17
Using a Single Key File for Multiple User-Configuration Files. 2-18
Examples. 2-18
Commands for Managing the Server Life Cycle 2-19
CANCEL_SHUTDOWN 2-20
Syntax 2-21
Example 2-21
DISCOVERMANAGEDSERVER. 2-21
Syntax 2-22
Example 2-23
FORCESHUTDOWN. 2-23
Syntax 2-24
Example 2-27
LOCK 2-28
Syntax 2-28
Example 2-28
RESUME 2-29
Syntax 2-29
Example 2-29
SHUTDOWN 2-30
Syntax 2-30
Example 2-34
START 2-35
Syntax 2-35
Example 2-36
STARTINSTANDBY 2-36
Syntax 2-37

Example. 2-38
UNLOCK 2-39
Syntax 2-39
Example. 2-39
Commands for Retrieving Information about WebLogic Server and Server Instances . 2-39
CONNECT 2-40
Syntax 2-41
Example. 2-41
GETSTATE 2-42
Syntax 2-42
Example. 2-43
HELP. 2-43
Syntax 2-43
Example. 2-43
LICENSES 2-44
Syntax 2-44
Example. 2-44
LIST 2-44
Syntax 2-45
Example. 2-45
PING 2-46
Syntax 2-46
Example. 2-46
SERVERLOG 2-47
Syntax 2-47
Example. 2-48
THREAD_DUMP. 2-49
Syntax 2-49

Example 2-50
VERSION 2-50
Syntax 2-50
Example 2-50
Commands for Managing JDBC Connection Pools 2-52
CREATE_POOL 2-53
Syntax 2-53
Example 2-55
DESTROY_POOL 2-56
Syntax 2-56
Example 2-56
DISABLE_POOL 2-57
Syntax 2-57
Example 2-58
ENABLE_POOL 2-58
Syntax 2-58
Example 2-58
TEST_POOL 2-59
Syntax 2-59
Example 2-60
RESET_POOL 2-60
Syntax 2-60
Example 2-60
EXISTS_POOL 2-61
Syntax 2-61
Example 2-61
Commands for Managing WebLogic Server MBeans 2-61
Specifying MBean Types 2-62

MBean Management Commands 2-62
CREATE 2-63
Syntax 2-64
Example. 2-64
DELETE 2-65
Syntax 2-65
Example. 2-66
GET 2-67
Syntax 2-67
Example. 2-68
INVOKE 2-69
Syntax 2-69
Example. 2-70
QUERY 2-71
Syntax 2-72
Example. 2-73
SET 2-74
Syntax 2-74
Example. 2-76
Running Commands in Batch Mode 2-77
BATCHUPDATE 2-77
Syntax 2-78
Example. 2-79
Commands for Working with Clusters 2-79
CLUSTERSTATE 2-80
Syntax 2-80
Example. 2-81
MIGRATE. 2-81

Syntax 2-82
Examples. 2-83
STARTCLUSTER. 2-83
Syntax 2-84
Example 2-84
STOPCLUSTER. 2-85
Syntax 2-85
Example 2-85
VALIDATECLUSTERCONFIG. 2-86
Syntax 2-86
Example 2-87

3. Using the WebLogic Server Java Utilities

appc. 3-3
AppletArchiver 3-3
Syntax 3-3
autotype (deprecated) 3-4
BuildXMLGen 3-4
CertGen. 3-4
Syntax 3-4
Example 3-6
ClientDeployer 3-7
clientgen 3-7
Conversion (deprecated). 3-7
dbping 3-7
Creating a DB2 Package with dbping. 3-8
Syntax 3-8
Example 3-10

ddcreate 3-11
DDInit 3-11
Deployer 3-12
der2pem 3-12
Syntax 3-12
Example 3-13
ejbc (deprecated) 3-13
EJBGen 3-14
encrypt 3-14
Syntax 3-14
Examples 3-15
getProperty 3-15
Syntax 3-15
Example 3-15
host2ior 3-16
Syntax 3-16
ImportPrivateKey 3-16
Syntax 3-16
Example 3-17
jhtml2jsp 3-18
Syntax 3-19
jspc (deprecated) 3-19
logToZip 3-19
Syntax 3-19
Examples 3-20
MBean Commands 3-20
MulticastTest 3-20
Syntax 3-21

Example 3-22
myip 3-23
Syntax 3-23
Example 3-23
pem2der 3-23
Syntax 3-23
Example 3-23
pointbase 3-24
rmic 3-24
Schema 3-24
Syntax 3-24
Example 3-25
servicegen (deprecated) 3-25
SearchAndBuild 3-25
Example 3-26
showLicenses 3-26
Syntax 3-26
Example 3-26
source2wsdd (deprecated). 3-26
system 3-27
Syntax 3-27
Example 3-27
ValidateCertChain 3-27
verboseToZip 3-28
Syntax 3-28
Example 3-29
wlappc 3-29
wlcompile 3-29

wlconfig 3-29
wldeploy 3-29
wlpackage 3-29
wlserver 3-30
writeLicense 3-30
Syntax 3-30
Examples 3-30
wsdl2Service 3-32
wsdlgen (deprecated) 3-32
wspackage (deprecated) 3-33

4. weblogic.Server Command-Line Reference

Required Environment and Syntax for weblogic.Server 4-1
Environment 4-2
Modifying the Classpath 4-2
Syntax 4-3
Default Behavior 4-3
weblogic.Server Configuration Options 4-4
JVM Parameters 4-5
Location of License and Configuration Data 4-6
Examples 4-8
Options that Override a Server's Configuration 4-9
Server Communication 4-10
SSL 4-15
Security 4-18
Message Output and Logging 4-24
Other Server Configuration Options 4-25
Clusters 4-28

Using the weblogic.Server Command Line to Start a Server Instance 4-28
Using the weblogic.Server Command Line to Create a Domain 4-29
Verifying Attribute Values That Are Set on the Command Line 4-31

5. WebLogic SNMP Agent Command-Line Reference

Required Environment for the SNMP Command-Line Interface 5-2
Syntax and Common Arguments for the SNMP Command-Line Interface 5-2
Commands for Retrieving WebLogic Server Managed Objects. 5-4
snmpwalk 5-4
Syntax 5-4
Example 5-5
snmpgetnext 5-6
Syntax 5-6
Example 5-6
snmpget. 5-8
Syntax 5-8
Example 5-8
Commands for Testing Traps 5-9
snmptrapd 5-10
Syntax 5-10
Example 5-10
snmpv1trap 5-10
Syntax 5-11
Example 5-12
Example: Using snmpv1trap to Send Traps to the Trap Daemon 5-13
Example: Using the WebLogic SNMP Agent to Send Traps to the Trap Daemon . 5-13

Index

CHAPTER 1

Introduction and Roadmap

This section describes the contents and organization of this guide—*WebLogic Server Command Reference*.

- "Document Scope and Audience" on page 1-1
- "Guide to This Document" on page 1-1
- "Related Documentation" on page 1-2

Document Scope and Audience

This document describes BEA WebLogic Server® command-line reference features and Java utilities and how to use them to administer WebLogic Server.

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.

Guide to This Document

The document is organized as follows:

- This chapter, "Introduction and Roadmap," describes the scope of this guide and lists related documentation.

- Chapter 2, “weblogic.Admin Command-Line Reference (Deprecated),” describes using the `weblogic.Admin` command to configure a WebLogic Server domain from a command shell or a script. Because the `weblogic.Admin` utility is deprecated in WebLogic Server 9.0, BEA Systems recommends that you use the WebLogic Scripting Tool (WLST) for equivalent functionality.
- Chapter 3, “Using the WebLogic Server Java Utilities,” describes various Java utilities you can use to manage and troubleshoot a WebLogic Server domain.
- Chapter 4, “weblogic.Server Command-Line Reference,” describes how to start WebLogic Server instances from a command shell or from a script.
- Chapter 5, “WebLogic SNMP Agent Command-Line Reference,” describes using Simple Network Management Protocol (SNMP) to communicate with enterprise-wide management systems.

Related Documentation

- "Using Ant Tasks to Configure a WebLogic Server Domain" in *Developing Applications with WebLogic Server*
- *WebLogic Scripting Tool*
- *Configuring WebLogic Server Environments*
- *Administration Console Online Help*

CHAPTER 2

weblogic.Admin Command-Line Reference (Deprecated)

Note: The `weblogic.Admin` utility is deprecated as of WebLogic Server® 9.0. Both `weblogic.Admin` utility and `wlconfig` tool are now restricted as follows:

- No access to MBeans that have been added in WebLogic Server 9.0. These tools use the compatibility MBean server to access MBeans, and this MBean server does not contain MBeans that are new in WebLogic Server 9.0.
- No longer able to configure security MBeans, but can still be used to view and invoke methods on the security MBeans.
- No longer able to create or modify Local Configuration MBeans, but can be used to view them and invoke their operations.

BEA Systems recommends that you use the WebLogic Scripting Tool (WLST) for equivalent functionality. For more information, see *WebLogic Scripting Tool*.

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

Like the Administration Console, for most commands 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 times from a shell script, consider using the `BATCHUPDATE` command, which is described in "Running Commands in Batch Mode" on page 2-77.

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

- "Required Environment for the weblogic.Admin Utility" on page 2-2
- "Syntax for Invoking the weblogic.Admin Utility" on page 2-3
- "Command for Storing User Credentials" on page 2-15
- "Commands for Managing the Server Life Cycle" on page 2-19
- "Commands for Retrieving Information about WebLogic Server and Server Instances" on page 2-39
- "Commands for Managing JDBC Connection Pools" on page 2-52
- "Commands for Managing WebLogic Server MBeans" on page 2-61
- "Running Commands in Batch Mode" on page 2-77
- "Commands for Working with Clusters" on page 2-79

For more information, see:

- "Automating WebLogic Server Administration Tasks" in *WebLogic Scripting Tool*. Describes using WLST commands to automate typical domain and server configuration tasks.
- "Using Ant Tasks to Configure a WebLogic Server Domain" in *Developing Applications with WebLogic Server.*
- "Overview of Deployment Tools" in *Deploying WebLogic Server Applications*. Describes deployment tools to help you configure and deploy applications.

Required Environment for the weblogic.Admin Utility

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/docs91/install/index.html`.
2. Add WebLogic Server classes to the `CLASSPATH` environment variable and *`WL_HOME`*`\server\bin` to the `PATH` environment variable.

   You can use a *`WL_HOME`*`\server\bin\setWLSEnv` script to set both variables. See "Modifying the Classpath" on page 4-2.
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 "Configure 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, see "SSL Arguments" on page 2-4.

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, see "Configure the domain-wide administration port" in the *Administration Console Online Help*.

Syntax for Invoking the weblogic.Admin Utility

```
java [ SSL Arguments ]
     weblogic.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     COMMAND-NAME command-arguments
```

The command names and arguments are not case sensitive.

The following sections provide detailed syntax information:

- "SSL Arguments" on page 2-4
- "Connection Arguments" on page 2-7
- "User Credentials Arguments" on page 2-10
- "Protocol Support" on page 2-13

Note: Both the weblogic.Deployer tool and the BEA WebLogic Scripting Tool (WLST) also use the SSL arguments, Connection arguments, and User Credentials arguments.

SSL Arguments

```
java [ -Dweblogic.security.TrustKeyStore=DemoTrust ]
     [ -Dweblogic.security.JavaStandardTrustKeyStorePassPhrase=password ]
     [ -Dweblogic.security.CustomTrustKeyStoreFileName=filename
       -Dweblogic.security.TrustKeyStoreType=CustomTrust
       [-Dweblogic.security.CustomTrustKeyStorePassPhrase=password ]
     ]
     [ -Dweblogic.security.SSL.hostnameVerifier=classname ]
     [ -Dweblogic.security.SSL.ignoreHostnameVerification=true ]
     weblogic.Admin
     [ User Credentials Arguments ]
     COMMAND-NAME command-arguments
```

If you have enabled the domain-wide administration port, or if you want to secure your administrative request by using some other listen port that is secured by SSL, you must include SSL arguments when you invoke `weblogic.Admin`. Table 2-1 describes all SSL arguments for the `weblogic.Admin` utility.

Table 2-1 SSL Arguments

Argument	Definition
`-Dweblogic.security.TrustKeyStore=DemoTrust`	Causes `weblogic.Admin` to trust the CA certificates in the demonstration trust keystore (`WL_HOME\server\lib\DemoTrust.jks`). This argument is required if the server instance to which you want to connect is using the demonstration identity and certificates. By default, `weblogic.Admin` trusts only the CA certificates in the Java Standard Trust keystore (`JAVA_HOME\jre\lib\security`).
`-Dweblogic.security.JavaStandardTrustKeystorePassPhrase=`*password*	Specifies the password that was used to secure the Java Standard Trust keystore. If the Java Standard Trust keystore is protected by a password, and if you want to trust its CA certificates, you must use this argument. By default, the Java Standard Trust keystore is not protected by a password.

Table 2-1 SSL Arguments

Argument	Definition
`-Dweblogic.security.CustomTrustKeyStoreFileName=`*`filename`* `-Dweblogic.security.TrustKeyStoreType=CustomTrust`	Causes `weblogic.Admin` to trust the CA certificates in a custom keystore that is located at *`filename`*. You must use both arguments to trust custom keystores.
`-Dweblogic.security.CustomTrustKeyStorePassPhrase=`*`password`*	Specifies the password that was used to secure the custom keystore. You must use this argument only if the custom keystore is protected by a password.
`-Dweblogic.security.SSL.hostnameVerifier=`*`classname`*	Specifies the name of a custom Host Name Verifier class. The class must implement the `weblogic.security.SSL.HostnameVerifier` interface.
`-Dweblogic.security.SSL.ignoreHostnameVerification=true`	Disables host name verification.

Using SSL to Secure Administration Requests: Main Steps

To secure administration requests with SSL:

1. Ensure that two-way SSL is disabled on the server instance to which you want to connect.

   By default, when you enable SSL, a server instance supports one-way SSL. Because two-way SSL provides additional security, you might have enabled two-way SSL. However, `weblogic.Admin` does not support two-way SSL.

2. Ensure that the trusted CA certificates are stored in a keystore that the `weblogic.Admin` utility can access through the file system.

3. When you invoke the `weblogic.Admin` utility, include arguments that specify the following:

   – A secure protocol and port.

   See "Protocol Support" on page 2-13.

   – (Optional) The trusted CA certificates and certificate authorities.

   See "Specifying Trust for weblogic.Admin" on page 2-6.

– (Optional) A host name verifier.

See "Specifying Host Name Verification for weblogic.Admin" on page 2-7.

Specifying Trust for weblogic.Admin

When the `weblogic.Admin` utility connects to a server's SSL port, it must specify a set of certificates that describe the certificate authorities (CAs) that the utility trusts.

To specify trust for `weblogic.Admin`:

- To trust only the CA certificates in the Java Standard Trust keystore, you do not need to specify command-line arguments, unless the keystore is protected by a password.

  If the Java Standard Trust keystore is protected by a password, use the following command-line argument:

  `-Dweblogic.security.JavaStandardTrustKeyStorePassPhrase=`*`password`*

- To trust both the CA certificates in the Java Standard Trust keystore **and** in the demonstration trust keystore, include the following argument:

  `-Dweblogic.security.TrustKeyStore=DemoTrust`

  This argument is required if the server instance to which you want to connect is using the demonstration identity and certificates.

  If the Java Standard Trust keystore is protected by a password, include the following command-line argument:

  `-Dweblogic.security.JavaStandardTrustKeyStorePassPhrase=`*`password`*

- To trust only the CA certificates in a keystore that you create, specify the following command-line arguments:

  – `-Dweblogic.security.CustomTrustKeyStoreFileName=`*`filename`*

    where *`filename`* specifies the fully qualified path to the trust keystore.

  – `-Dweblogic.security.TrustKeyStoreType=CustomTrust`

    This optional command-line argument specifies the type of the keystore. Generally, this value for type is `jks`.

  – If the custom keystore is protected by a password, include `-Dweblogic.security.CustomTrustKeyStorePassPhrase=`*`password`*

Specifying Host Name Verification for weblogic.Admin

A host name verifier ensures the host name URL to which the client connects matches the host name in the digital certificate that the server sends back as part of the SSL connection. A host name verifier is useful when an SSL client, or a SSL server acting as a client, connects to an application server on a remote host. It helps to prevent man-in-the-middle attacks. See "Using Host Name Verification" in *Managing WebLogic Security*.

To specify host name verification for `weblogic.Admin`:

- To use the host name verifier that the WebLogic Security Service provides, you do not need to specify host name verification arguments.

  Note: If you specify an IP address or the `localhost` string in the `weblogic.Admin -url` or `-adminurl` argument, the host name verifier that the WebLogic Security Service provides will allow the connection if the common name (CN) field of the digital certificate matches the DNS name of the local host.

- To use a custom host name verifier, specify:
  `-Dweblogic.security.SSL.hostnameVerifier=`*`classname`*

  where *`classname`* specifies the implementation of the `weblogic.security.SSL.HostnameVerifier` interface.

- To disable host name verification, specify:
  `-Dweblogic.security.SSL.ignoreHostnameVerification=true`

Connection Arguments

```
java [ SSL Arguments ]
     weblogic.Admin
     [ {-url URL} | {-adminurl URL} ]
     [ User Credentials Arguments ]
     COMMAND-NAME command-arguments
```

When you invoke most `weblogic.Admin` commands, you specify the arguments in Table 2-2 to connect to a WebLogic Server instance. Some commands have special requirements for the connection arguments. Any special requirements are described in the command documentation.

Table 2-2 Connection 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 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, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities with JMX*. 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*, see "Protocol Support" on page 2-13. For more information about the listen address and listen ports, see "-Dweblogic.ListenAddress=host" on page 4-13 and "-Dweblogic.ListenPort= portnumber" on page 4-13. For more information about the domain-wide administration port, see "Configure the domain-wide administration port" in the *Administration Console Online Help*. The default value for this argument is `t3://localhost:7001`.

Table 2-2 Connection Arguments (Continued)

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, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities with JMX*. For all commands other than the MBean commands, `-adminurl admin-address` and `-url admin-address` are synonymous. 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*, see "Protocol Support" on page 2-13. There is no default value for this argument.

User Credentials Arguments

```
java [ SSL Arguments ]
     weblogic.Admin
     [ Connection Arguments ]
     [ { -username username [-password password] } |
       [ -userconfigfile config-file [-userkeyfile admin-key] ]
     ]
     COMMAND-NAME command-arguments
```

When you invoke most `weblogic.Admin` commands, you specify the arguments in Table 2-3 to provide the user credentials of a WebLogic Server user who has permission to invoke the command.

Table 2-3 User Credentials Arguments

Argument	Definition
`-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, see "Users, Groups, And Security Roles" in *Securing WebLogic Resources*.
`-password` *`password`*	The password that is associated with the username. If you do not specify the `-password` argument, `weblogic.Admin` prompts you for a password. If *`WL_HOME`*`\server\bin` is specified in the `PATH` environment variable, `weblogic.Admin` uses a set of WebLogic Server libraries that prevent the password from being echoed to standard out. For information on setting environment variables, see "Required Environment for the weblogic.Admin Utility" on page 2-2.
`-userconfigfile` *`config-file`*	Specifies the name and location of a user-configuration file, which contains an encrypted username and password. The encrypted username must have permission to invoke the command you specify. If you do not specify `-userconfigfile` *`config-file`*, and if you do not specify `-username` *`username`*, `weblogic.Admin` searches for a user-configuration file at the default path name. (See "STOREUSERCONFIG" on page 2-15.)
`-userkeyfile` *`admin-key`*	Specifies the name and location of the key file that is associated with the user-configuration file you specify. When you create a user-configuration file, the `STOREUSERCONFIG` command uses a key file to encrypt the username and password. Only the key file that encrypts a user-configuration file can decrypt the username and password. If you do not specify `-userkeyfile` *`admin-key`*, `weblogic.Admin` searches for a key file at the default path name. (See "STOREUSERCONFIG" on page 2-15.)

Note: The exit code for all commands is 1 if the Administration client cannot connect to the server or if the WebLogic Server instance rejects the username and password.

Specifying User Credentials

The simplest way to specify user credentials is to create a user configuration file and key file in the default location. Thereafter, you do not need to include user credentials in `weblogic.Admin` invocations. A user-configuration file contains encrypted user credentials that can be decrypted only by a single key file. See "STOREUSERCONFIG" on page 2-15.

For example, the following command creates a user configuration file and key file in the default location:

```
java weblogic.Admin -username weblogic -password weblogic STOREUSERCONFIG
```

After you enter this `STOREUSERCONFIG` command, you can invoke `weblogic.Admin` without specifying credentials on the command line or in scripts. For example:

```
java weblogic.Admin GET -pretty -type -Domain
```

If you create a user configuration file or key file in a location other than the default, you can include the `-userconfigfile` *`config-file`* and `-userkeyfile` *`admin-key`* arguments on the command line or in scripts.

If you do not create a user configuration file and key file, you must use the `-username` and `-password` arguments when invoking the `weblogic.Admin` utility directly on the command line or in scripts. With these arguments, the username and password are not encrypted. If you store the values in a script, the user credentials can be used by anyone who has read access to the script.

The following list summarizes the order of precedence for the `weblogic.Admin` user-credentials arguments:

- If you specify `-username` *`username`* `-password` *`password`*, the utility passes the unencrypted values to the server instance you specify in the `-url` argument.
- These arguments take precedence over the `{ -userconfigfile` *`config-file`* `-userkeyfile` *`admin-key`* `}` arguments.
- If you specify `-username` *`username`*, the utility prompts for a password. Then it passes the unencrypted values to the server instance you specify in the `-url` argument.
- This argument also takes precedence over the `{ -userconfigfile` *`config-file`* `-userkeyfile` *`admin-key`* `}` arguments.
- If you specify `{ -userconfigfile` *`config-file`* `-userkeyfile` *`admin-key`* `}` and do not specify `{ -username` *`username`* `[-password` *`password`*`]}`, the utility passes the values that are encrypted in *`config-file`* to the server instance you specify in the `-url` argument.

- If you specify neither `{ -username username [-password password] }` nor `{ -userconfigfile config-file -userkeyfile admin-key }`, the utility searches for a user-configuration file and key file at the default path names. The default path names vary depending on the JVM and the operating system. See "Configuring the Default Path Name" on page 2-17.

Examples of Providing User Credentials

The following command specifies the username **weblogic** and password **weblogic** directly on the command line:

```
java weblogic.Admin -username weblogic -password weblogic COMMAND
```

The following command uses a user-configuration file and key file that are located at the default pathname:

```
java weblogic.Admin COMMAND
```

See "Configuring the Default Path Name" on page 2-17.

The following command uses a user-configuration file named `c:\wlUser1-WebLogicConfig.properties` and a key file named `e:\secure\myKey`:

```
java -userconfigfile c:\wlUser1-WebLogicConfig.properties
-userkeyfile e:\secure\myKey COMMAND
```

Protocol Support

The `-url` and `-adminurl` arguments of the `weblogic.Admin` utility support the `t3`, `t3s`, `http`, `https`, and `iiop` 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, see "Configure HTTP Protocol" in the *Administration Console Online Help*.

If you want to use `iiop` to connect to a server instance, you must enable the `iiop` protocol for that instance. For more information, see "Enable and Configure IIOP" 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`. See "Using SSL to Secure Administration Requests: Main Steps" on page 2-5.

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 7011.
- 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, see "-Dweblogic.ListenAddress=host" on page 4-13 and "-Dweblogic.ListenPort= portnumber" on page 4-13.
- The `weblogic` username has system-administrator privileges and uses `weblogic` for a password.
- The user credentials have **not** been encrypted in a user configuration file.
- 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
```

`weblogic.Admin` calls `System.exit(1)` if an exception is raised while processing a command, causing Ant and other Java client JVMs to exit. You can override this default behavior by specifying `-noExit` for Ant tasks (`wlconfig`) and `-continueOnError` for `weblogic.Admin` batch operations (`BATCHUPDATE`).

Command for Storing User Credentials

For any `weblogic.Admin` command that connects to a WebLogic Server instance, you must provide user credentials. You can use the `STOREUSERCONFIG` command to encrypt the user credentials instead of passing credentials directly on the command line or storing unencrypted credentials in scripts. See "Specifying User Credentials" on page 2-12.

STOREUSERCONFIG

Creates a user-configuration file and an associated key file. The user-configuration file contains an encrypted username and password. The key file contains a secret key that is used to encrypt and decrypt the username and password.

Only the key file that originally encrypted the username and password can decrypt the values. If you lose the key file, you must create a new user-configuration and key file pair.

Caution: You must ensure that only authorized users can access the key file. Any user who accesses a valid user-configuration and key file pair gains the privileges of the encrypted username. To secure access to the key file, you can store the key file in a directory that provides read and write access only to authorized users, such as WebLogic Server administrators. Alternatively, you can write the key file to a removable medium, such as a floppy or CD, and lock the medium in a drawer when it is not being used.

Unlike other `weblogic.Admin` commands, the `STOREUSERCONFIG` command does not connect to a WebLogic Server instance. The data encryption and file creation are accomplished by the JVM in which the `STOREUSERCONFIG` command runs. Because it does not connect to a WebLogic Server instance, the command cannot verify that the username and password are valid WebLogic Server credentials.

Syntax

```
java weblogic.Admin
  -username username [-password password]
  [ -userconfigfile config-file ] [ -userkeyfile keyfile ]
  STOREUSERCONFIG
```

Argument	Definition
`-userconfigfile` *`config-file`*	Specifies a file pathname at which the `STOREUSERCONFIG` command creates a user-configuration file. The pathname can be absolute or relative to the directory from which you enter the command. If a file already exists at the specified pathname, the command overwrites the file with a new file that contains the newly encrypted username and password. If you do not specify this option, `STOREUSERCONFIG` does the following: • To determine the directory in which to create the user-configuration file, it uses the JVM's user-home directory. The default value varies depending on the SDK and type of operating system. See "Configuring the Default Path Name" on page 2-17. • To determine the file name, it prepends your operating-system username to the string `-WebLogicConfig.properties`. For example, *`username`*`-WebLogicConfig.properties`. You can use Java options to specify a different username. See "Configuring the Default Path Name" on page 2-17.
`-userkeyfile` *`keyfile`*	Specifies a file pathname at which the `STOREUSERCONFIG` command creates a key file. The pathname can be absolute or relative to the directory from which you enter the command. If a file already exists at the specified pathname, `STOREUSERCONFIG` uses the existing key file to encrypt the new user-configuration file. If you do not specify this option, `STOREUSERCONFIG` does the following: • To determine the directory in which to create the key file, it uses the JVM's user-home directory. The default value varies depending on the SDK and type of operating system. See "Configuring the Default Path Name" on page 2-17. • To determine the file name, it prepends your operating-system username to the string `-WebLogicConfig.properties`. For example, *`username`*`-WebLogicConfig.properties`. You can use Java options to specify a different username. See "Configuring the Default Path Name" on page 2-17.
`-username` *`username`* `[-password` *`password`* `]`	Specifies the username and password to encrypt. The `STOREUSERCONFIG` command does **not** verify that the username and password are valid WebLogic Server user credentials. If you omit the `-password` *`password`* argument, `STOREUSERCONFIG` prompts you to enter a password.

Configuring the Default Path Name

If you do not specify the location in which to create and use a user-configuration file and key file, the `weblogic.Admin` and `weblogic.Deployer` utilities supply the following default values:

- *`user-home-directory`*`\`*`username`*`-WebLogicConfig.properties`
- *`user-home-directory`*`\`*`username`*`-WebLogicKey.properties`

Where *`user-home-directory`* is the home directory of the operating-system user account as determined by the JVM, and *`username`* is your operating-system username.

The value of the home directory varies depending on the SDK and type of operating system. For example, on UNIX, the home directory is usually "*`~username`*." On Windows, the home directory is usually "`C:\Documents and Settings\`*`username`*".

You can use the following Java options to specify values for *`user-home-directory`* and *`username`*:

- `-Duser.home=`*`pathname`* specifies the value of *`user-home-directory`*
- `-Duser.name=`*`username`* specifies the value of *`username`*.

For example, the following command configures the user-home directory to be `c:\myHome` and the user name to be `wlAdmin`. The command will search for the following user-configuration file and user key file:

```
c:\myHome\wlAdmin-WebLogicConfig.properties
c:\myHome\wlAdmin-WebLogicKey.properties
```

```
java -Duser.home=c:\myHome -Duser.name=wlAdmin
weblogic.Admin COMMAND
```

Creating User-Configuration and Key Files

To create user-configuration and key files:

1. Use the `-username` *`username`* and `-password` *`password`* arguments to specify the username and password to be encrypted.
2. Specify the name and location of the user-configuration and key files by doing one of the following:
   - Use the `-userconfigfile` *`config-file`* and `-userkeyfile` *`key-file`* arguments:
     ```
     java weblogic.Admin -username username -password password
      -userconfigfile config-file -userkeyfile key-file
     STOREUSERCONFIG
     ```

  - Use the default behavior to create files named *user-home-directory**username*-WebLogicConfig.properties and *user-home-directory**username*-WebLogicKey.properties:
    ```
    java weblogic.Admin -username username -password password
    STOREUSERCONFIG
    ```
  - Use the -Duser.home=*directory* and -Duser.name=*username* Java options to create files named *directory**username*-WebLogicConfig.properties and *directory**username*-WebLogicKey.properties:
    ```
    java -Duser.home=directory -Duser.name=username
    weblogic.Admin -username username -password password
    STOREUSERCONFIG
    ```

You can change the name and location of a user-configuration file or a key file after you create them, as long as you use the two files as a pair.

Using a Single Key File for Multiple User-Configuration Files

To use one key file to encrypt multiple user-configuration files:

1. Create an initial user-configuration file and key file pair.

   For example, enter the following command:

   ```
   java weblogic.Admin -username username -password password
   -userconfigfile c:\AdminConfig -userkeyfile e:\myKeyFile
   STOREUSERCONFIG
   ```

2. When you create an additional user-configuration file, specify the existing key file.

   For example, enter the following command:

   ```
   java weblogic.Admin -username username -password password
   -userconfigfile c:\anotherConfigFile -userkeyfile e:\myKeyFile
   STOREUSERCONFIG
   ```

Examples

In the following example, a user who is logged in to a UNIX operating system as `joe` encrypts the username `wlAdmin` and password `wlPass`:

```
java weblogic.Admin -username wlAdmin -password wlPass
STOREUSERCONFIG
```

The command determines whether a key file named `~joe/joe-WebLogicKey.properties` exists. If such a file does not exist, it prompts the user to select `y` to confirm creating a key file.

If the command succeeds, it creates two files:

```
~joe\joe-WebLogicConfig.properties
~joe\joe-WebLogicKey.properties
```

The file `joe-WebLogicConfig.properties` contains an encrypted version of the strings `wlAdmin` and `wlPass`. Any command that uses the `~joe\joe-WebLogicConfig.properties` file must specify the `~joe\joe-WebLogicKey.properties` key file.

In the following example, the user `joe` is a System Administrator who wants to create a user-configuration file for an operating-system account named `pat`. For the sake of convenience, `joe` wants to create the user-configuration file in `pat`'s home directory, which will simplify the syntax of the `weblogic.Admin` commands that `pat` invokes. For added security, only one key file exists at `joe`'s organization, and it is located on a removable hard drive.

To create a user configuration file in `pat`'s home directory that is encrypted and decrypted by a key file name `e:\myKeyFile`:

```
java -Duser.name=pat -Duser.home="C:\Documents and Settings\pat"
weblogic.Admin -username wlOperatorPat -password wlOperator1 -userkeyfile
e:\myKeyFile
STOREUSERCONFIG
```

A user who logs in to `pat`'s account can use the following syntax to invoke `weblogic.Admin` commands:

```
java weblogic.Admin -userkeyfile e:\myKeyFile COMMAND
```

For information on using user-configuration and key files, see "Specifying User Credentials" on page 2-12.

Commands for Managing the Server Life Cycle

Table 2-4 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, see "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown* and "Managing Servers and Server Life Cycle" in *WebLogic Scripting Tool*.

Table 2-4 Overview of Commands for Managing the Server Life Cycle

Command	Description
CANCEL_SHUTDOWN	Cancels the SHUTDOWN command for the WebLogic Server that is specified in the URL. See "CANCEL_SHUTDOWN" on page 2-20.
DISCOVERMANAGEDSERVER	Causes the Administration Server to re-establish its administrative control over Managed servers. See "DISCOVERMANAGEDSERVER" on page 2-21.
FORCESHUTDOWN	Terminates a server instance without waiting for active sessions to complete. See "FORCESHUTDOWN" on page 2-23.
LOCK	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 2-28.
RESUME	Makes a server available to receive requests from external clients. See "RESUME" on page 2-29.
SHUTDOWN	Gracefully shuts down a WebLogic Server. See "SHUTDOWN" on page 2-30.
START	Uses a configured Node Manager to start a Managed Server in the RUNNING state. See "START" on page 2-35.
STARTINSTANDBY	Uses a configured Node Manager to start a Managed Server and place it in the STANDBY state. See "STARTINSTANDBY" on page 2-36.
UNLOCK	Unlocks the specified WebLogic Server after a LOCK operation. See "UNLOCK" on page 2-39.

CANCEL_SHUTDOWN

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 2-30 and "UNLOCK" on page 2-39.

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, see "Controlling Graceful Shutdowns" in *Managing Server Startup and Shutdown*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     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, see "Example Environment" on page 2-14.

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. By default, an Administration Server finds the last known set of Managed Servers and re-establishes a connection.

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

For example, you might have started a Managed Server in the `STANDBY` state and did not resume the Managed Server before restarting the Administration Server. The Administration Server discovers only Managed Servers that are in the `RUNNING` state.

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.

In WebLogic Server 9.0, this command is deprecated because if an Administration Server stops running while the Managed Servers in the domain continue to run, each Managed Server will periodically attempt to reconnect to the Administration Server at the interval specified by the `ServerMBean` attribute `AdminReconnectIntervalSecs`.

For more information, see "Managed Servers and Re-started Administration Server" in *Managing Server Startup and Shutdown*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [ -url [protocol://]listen-address:listen-port ]
     [ User Credentials Arguments ]
     DISCOVERMANAGEDSERVER [-serverName targetServer
     [-listenAddress listenaddress] [-listenPort listenport]]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	You must 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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`-serverName targetServer`	Specifies 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.

Argument	Definition (Continued)
`-listenAddress` *`listenaddress`*	Specifies the listen address of the Managed Server that you name with the `-serverName` argument. If you do not specify this argument, the command uses the listen address that is configured in the domain's `config.xml` file.
`-listenPort` *`listenport`*	Specifies the listen port of the Managed Server that you name with the `-serverName` argument. If you do not specify this argument, the command uses the listen port that is configured in the domain's `config.xml` file.

Example

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

```
java weblogic.Admin -url AdminHost:7011 -username weblogic
   -password weblogic DISCOVERMANAGEDSERVER
   -serverName MedRecManagedServer
```

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

In the following example, a Managed Server is configured to listen on port 7021, but you used the `-Dweblogic.ListenPort` startup option to temporarily change the listen port to `8201`. The following command instructs the Administration Server to re-connect to MedRecManagedServer, which is listening on port 8201:

```
java weblogic.Admin -url AdminHost:7011 -username weblogic
   -password weblogic DISCOVERMANAGEDSERVER
   -serverName MedRecManagedServer -listenPort 8201
```

FORCESHUTDOWN

Terminates a server instance without waiting for active sessions to complete. For more information, see "Force Shutdown" in *Managing Server Startup and Shutdown*.

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, see "Configure the domain-wide administration port" in the *Administration Console Online Help*.

Syntax

```
java [ SSL Arguments ]
   [-Dweblogic.system.BootIdentityFile=filename
     [-Dweblogic.RootDirectory=path]
   ]
   weblogic.Admin
   [ Connection Arguments ]
   [ User Credentials Arguments ]
   FORCESHUTDOWN [targetServer]
```

Argument	Definition
`-Dweblogic.system.BootIdentityFile=`*`filename`* `[-Dweblogic.RootDirectory=`*`path`*`]`	Cause the command to retrieve encrypted user credentials from a boot identity file. See "Boot Identity Files" in *Managing Server Startup and Shutdown*. Use these arguments if you invoke this command from a script, you have not created a user configuration file, and you do not want to store user credentials in your script. If you do you not use the `-username` argument or a user configuration file to specify credentials (see "User Credentials Arguments" on page 2-10), the command retrieves user credentials from a boot properteis file as follows: • If you invoke the command from a server's root directory, and if the server's root directory contains a valid `boot.properties` file, it retrieves credentials from this file by default. For information about a server's root directory, see "A Server's Root Directory." • If you invoke the command from a server's root directory, but the server's boot identity file is not in the server's root directory or is not named `boot.properties`, the command can use a boot identity file if you include the following argument: `-Dweblogic.system.BootIdentityFile=`*`filename`* where *`filename`* is the fully qualified pathname of a valid boot identity file. • If you do not invoke the command from a server's root directory, the command can use a boot identity file if you include both of the following arguments in the command: `-Dweblogic.system.BootIdentityFile=`*`filename`* `-Dweblogic.RootDirectory=`*`path`* where *`filename`* is the fully qualified pathname of a valid boot identity file and `path` is the relative or fully-qualified name of the server's root directory. • If you have not created a boot identity file for a server, or if you do not want to use it, you must use the `-username` and `-password` arguments to provide user credentials. • If you specify both `-Dweblogic.system.BootIdentityFile=`*`filename`* and `-username` and `-password`, the command uses the credentials specified in the `-username` and `-password` arguments.

Argument	Definition (Continued)
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:7011 -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, see "Example Environment" on page 2-14.

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
```

The following example provides user credentials by referring to a boot identity file. The example specifies the server's root directory and boot identity file name so that it can be invoked from any directory:

```
java -Dweblogic.system.BootIdentityFile=c:\mydomain\boot.properties
 -Dweblogic.RootDirectory=c:\mydomain
 weblogic.Admin -url AdminHost:7001 FORCESHUTDOWN
```

LOCK

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. See "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown*.

Syntax

```
java [ SSL Arguments ]
     weblogic.Admin
     [ -url [protocol://]listen-address:listen-port ]
     [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`"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` or `ADMIN` state to the `RUNNING` state.

For more information about server states, see "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown*.

Syntax

```
java [ SSL Arguments ]
     weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`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, see "Example Environment" on page 2-14.

SHUTDOWN

Gracefully shuts down the specified WebLogic Server instance.

A graceful shutdown gives WebLogic Server subsystems time to complete certain application processing currently in progress. By default, a server instance waits for pending HTTP sessions to finish as a part of the graceful shutdown. You can override this behavior using the `-ignoreExistingSessions` argument. See "Controlling Graceful Shutdowns" in *Managing Server Startup and Shutdown*.

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, see "Shut Down a Server Instance" in the *Administration Console Online Help*.

Instead of specifying a delay in the `SHUTDOWN` command, you can now use a `-timeout` option, or set attributes in the Administration Console to control how a server shuts down. For more information, see "Controlling Graceful Shutdowns" in *Managing Server Startup and Shutdown*.

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, see "Configure the domain-wide administration port" in the *Administration Console Online Help*.

Syntax

```
java [ SSL Arguments ]
     [-Dweblogic.system.BootIdentityFile=filename
       [-Dweblogic.RootDirectory=path]
     ]
```

```
      weblogic.Admin
      [ Connection Arguments ]
      [ User Credentials Arguments ]
    SHUTDOWN [-ignoreExistingSessions] [-timeout seconds]
    [targetServer]

(Deprecated)java [ SSL Arguments ]
      [-Dweblogic.system.BootIdentityFile=filename
        [-Dweblogic.RootDirectory=path]
      ]
      weblogic.Admin
      [ Connection Arguments ]
      [ User Credentials Arguments ]
    SHUTDOWN [seconds ["stringMessage"]] [targetServer]
```

Argument	Definition
`-Dweblogic.system.BootIdentityFile=`*`filename`* `[-Dweblogic.RootDirectory=`*`path`*`]`	Cause the command to retrieve encrypted user credentials from a boot identity file. See "Boot Identity Files" in *Managing Server Startup and Shutdown*. Use these arguments if you invoke this command from a script, you have not created a user configuration file, and you do not want to store user credentials in your script. If you do you not use the `-username` argument or a user configuration file to specify credentials (see "User Credentials Arguments" on page 2-10), the command retrieves user credentials from a boot properteis file as follows: • If you invoke the command from a server's root directory, and if the server's root directory contains a valid `boot.properties` file, it retrieves credentials from this file by default. For information about a server's root directory, see "A Server's Root Directory." • If you invoke the command from a server's root directory, but the server's boot identity file is not in the server's root directory or is not named `boot.properties`, the command can use a boot identity file if you include the following argument: `-Dweblogic.system.BootIdentityFile=`*`filename`* where *`filename`* is the fully qualified pathname of a valid boot identity file. • If you do not invoke the command from a server's root directory, the command can use a boot identity file if you include both of the following arguments in the command: `-Dweblogic.system.BootIdentityFile=`*`filename`* `-Dweblogic.RootDirectory=`*`path`* where *`filename`* is the fully qualified pathname of a valid boot identity file and `path` is the relative or fully-qualified name of the server's root directory. • If you have not created a boot identity file for a server, or if you do not want to use it, you must use the `-username` and `-password` arguments to provide user credentials. • If you specify both `-Dweblogic.system.BootIdentityFile=`*`filename`* and `-username` and `-password`, the command uses the credentials specified in the `-username` and `-password` arguments.

Argument	Definition
`-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, see "Controlling Graceful Shutdowns" in *Managing Server Startup and Shutdown*. By default, a graceful shutdown operation waits for HTTP sessions to complete or timeout.
`-timeout` *`seconds`*	The number of seconds subsystems have to complete in-flight work and suspend themselves. If you specify a timeout value and the subsystems do not complete work and suspend themselves within that period, WebLogic Server will perform a forced shutdown on the server instance.
`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.
`seconds`	Number of seconds allowed to elapse between the invoking of this command and the shutdown of the server.
`"stringMessage"`	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:7011 -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, see "Example Environment" on page 2-14.

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
```

The following example provides user credentials by referring to a boot identity file. The example specifies the server's root directory and boot identity file name so that it can be invoked from any directory:

```
java -Dweblogic.system.BootIdentityFile=c:\mydomain\boot.properties
 -Dweblogic.RootDirectory=c:\mydomain weblogic.Admin
 -url AdminHost:7001 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 must be configured to communicate with a Node Manager. For more information, see "Configure Machines" and "Assign Server Instances to Machines" in the *Administration Console Online Help*.

The Startup Mode field in the Administration Console determines whether a Managed Server starts in the `RUNNING`, `STANDBY`, or `ADMIN` state. See "Specify a Startup Mode" in the *Administration Console Online Help* and "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
   [-url URL]
   [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
targetServer	The name of the Managed Server to start.

Example

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

```
java weblogic.Admin -url AdminHost:7011 -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` in the state that the Startup Mode field specifies.
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 *ServerName*→Control→Remote Start Output page.
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, see "Example Environment" on page 2-14.

STARTINSTANDBY

Starts a Managed Server using Node Manager.

Prior to WebLogic Server 9.0, this command started a Managed Server using the Node Manager **and** placed it in a `STANDBY` state. In this state, a server is not accessible to requests from external clients.

In WebLogic Server 9.0, the Startup Mode field in the Administration Console determines the state in which a Managed Server starts, regardless of which command you use to start the server instance. See "Specify a Startup Mode" and "Start Managed Servers in the `STANDBY` Mode" in the *Administration Console Online Help*.

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 must be configured to communicate with a Node Manager. For more information, see "Configure Machines" and "Assign Server Instances to Machines" in the *Administration Console Online Help*.
- The domain must be configured to use a domain-wide administration port as described in "Configure the domain-wide administration port" in the *Administration Console Online Help*.

For more information about server states, see "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
   [-url URL]
   [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`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:

```
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` in the state that the Start Mode field specifies.
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 *`ServerName`*→Control→Remote Start Output page.
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 Machines→Monitoring→Node Manager Log page.

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

UNLOCK

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, see "RESUME" on page 2-29.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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 2-5 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 2-5 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 2-40.
GETSTATE	Returns the current state of the specified WebLogic Server instance. See "GETSTATE" on page 2-42.

Table 2-5 Overview of Commands for Retrieving Information about WebLogic Server (Continued)

Command	Description
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 2-43.
LICENSES	Lists the licenses for all WebLogic Server instances that are installed on a specific server. See "LICENSES" on page 2-44.
LIST	Lists the bindings of a node in a server's JNDI naming tree. See "LIST" on page 2-44.
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 2-46. For a similar command that returns information about all servers in a cluster, see "CLUSTERSTATE" on page 2-80.
SERVERLOG	Displays the server log file generated on a specific server instance. See "SERVERLOG" on page 2-47.
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 2-49.
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 2-50.

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 [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
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, see "Example Environment" on page 2-14.

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, see "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown*.

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, see "Configure the domain-wide administration port" in the *Administration Console Online Help*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
   GETSTATE [targetServer]
```

Argument	Definition
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, see "Example Environment" on page 2-14.

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

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

For a complete list of server states, see "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown*.

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
ic ping 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 [ SSL Arguments ] weblogic.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
   LICENSES
```

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, see "Example Environment" on page 2-14.

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 [ SSL Arguments ] weblogic.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
   LIST [JNDIcontextName]
```

Argument	Definition
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 the MedRecServer example server that runs on a machine named AdminHost:

```
java weblogic.Admin -url AdminHost:7011 -username weblogic
   -password weblogic LIST
```

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

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

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

To view the JNDI tree below the mail context, enter the following command:

```
java weblogic.Admin -url AdminHost:7011 -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, see "CLUSTERSTATE" on page 2-80.

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, see "Configure the domain-wide administration port" in the *Administration Console Online Help*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
   PING [roundTrips] [messageLength]
```

Argument	Definition
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, see "Example Environment" on page 2-14.

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.

By default, the command returns the first 500 messages from the current log file (messages within the file are ordered from oldest to newest). You can change the default behavior by specifying a time and date range, but you cannot change the number of messages to be returned. The command always returns up to 500 messages, depending on the number of messages in the log file.

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

```
MessageID TimeStamp Severity Subsystem MessageText
```

For more information about message attributes, see "Message Attributes" in the *Configuring Log Files and Filtering Log Messages*.

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, see "View and configure logs" in the *Administration Console Online Help*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
starttime	Returns up to 500 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 up to 500 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 up to 500 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 order 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, see "Example Environment" on page 2-14.

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, see "Configure the domain-wide administration port" in the *Administration Console Online Help*.

Note: The `THREAD_DUMP` command is supported only on Sun JVM and BEA JRockit®.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
     THREAD_DUMP
```

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, see "Example Environment" on page 2-14.

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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.

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, see "Example Environment" on page 2-14.

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

```
WebLogic Server 9.0  Sat Oct 15 22:51:04 EDT 2002 207896
WebLogic XMLX Module 9.0  Sat Oct 15 22:51:04 EDT 2002 207896
```

Commands for Managing JDBC Connection Pools

Table 2-6 lists the 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 "Simplified JDBC Resource Configuration" in *Programming WebLogic JDBC*.

Note: All JDBC commands are privileged commands. You must supply the username and password for a WebLogic Server administrative user to use these commands.

Table 2-6 Overview of Commands for Managing JDBC Connection Pools

Command	Description
`CREATE_POOL`	Creates a connection pool. See "CREATE_POOL" on page 2-53.
`DESTROY_POOL`	Closes all connections in the connection pool and deletes the connection pool configuration. See "DESTROY_POOL" on page 2-56.
`DISABLE_POOL`	Temporarily disables a connection pool, preventing any clients from obtaining a connection from the pool. See "DISABLE_POOL" on page 2-57.
`ENABLE_POOL`	Enables a connection pool 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 2-58.
`TEST_POOL`	Tests a connection pool by reserving and releasing a connection from the connection pool. Requires testConnsOnReserve or testConnsOnRelease to be true and testTableName must be set. See "TEST_POOL" on page 2-59

Table 2-6 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 2-60.
EXISTS_POOL	Tests whether a connection pool with a specified name exists in a specified 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 2-61.

CREATE_POOL

Creates a connection pool on the WebLogic Server instance running at the specified URL.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [ Connection Arguments ]
     [ User Credentials Arguments ]
   CREATE_POOL poolName props=myProps,initialCapacity=1,maxCapacity=1,
   capacityIncrement=1,allowShrinking=true,shrinkPeriodMins=15,
   driver=myDriver,url=myURL, testConnsOnReserve=true,
   testTableName=tablename
```

Argument	Definition
poolName	Required. A unique name of the connection pool. Must be unique in the domain.
aclName	Identifies the different access lists within `fileRealm.properties` in the server configuration directory. Paired name must be dynaPool.
props	Required. Database connection properties; typically in the format "database login name; server network id". Required entries vary by DBMS. Separate property value pairs with a semicolon.

Argument	Definition
password	Optional. Database login password. This value overrides any database password specified in props.
initialCapacity	Optional. Initial number of connections in a pool. If this property is defined, WebLogic Server creates these connections at boot time. Default is 1; cannot exceed maxCapacity.
maxCapacity	Optional. Maximum number of connections allowed in the pool. Default is 1; if defined, maxCapacity should be =>1.
capacityIncrement	Optional. Number of connections to add at a time when the connection pool is expanded. Default = 1.
allowShrinking	Optional. Indicates whether or not the pool can shrink when connections are detected to not be in use. Default = true.
shrinkPeriodMins	Optional. Interval between shrinking. Units in minutes. Minimum = 1. If `allowShrinking = True`, then default = 15 minutes.
driver	Required. Fully qualified driver classname of the JDBC driver.
url	Required. URL of the database as required by the JDBC driver. Format varies by DBMS.
testConnsOnReserve	Optional. Indicates reserved test connections. Default = False.
testConnsOnRelease	Optional. Indicates test connections when they are released. Default = False.
testTableName	Optional. Database table used when testing connections; must be present for tests to succeed. Required if either testConnsOnReserve or testConnsOnRelease are defined. Can also be a SQL query instead of a database table name. To use a SQL query, enter `SQL` followed by a space and the SQL query to run in place of the standard test.

Argument	Definition
`refreshPeriod`	Optional. Sets the connection refresh interval in minutes. 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`	Optional. 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

The following example creates a connection pool named demoPool on the MedRecManagedServer instance running on the host machine named ManagedHost and listening at port 8001:

```
java weblogic.Admin -url t3://ManagedHost:8001 -username weblogic
   -password weblogic CREATE_POOL demoPool
   url=jdbc:pointbase:server://localhost:9092/demo,
   driver=com.pointbase.jdbc.jdbcUniversalDriver,
   testConnsOnReserve=true,
   testTableName=SYSTABLES,
   initialCapacity=2
   maxCapacity=10
   capacityIncrement=2
   allowShrinking=true
   props=user=examples;password=examples
```

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

```
Connection pool "demoPool" created successfully.
```

DESTROY_POOL

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

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
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

The following command destroys a connection pool named demoPool in a WebLogic Domain with the Administration Server running on a machine named AdminHost and listening at port 7001.

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic DESTROY_POOL demoPool false
```

DISABLE_POOL

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

You have the following options for disabling a pool. 1) Freeze the connections in a pool that you later plan to enable, or 2) Destroy the connections.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   DISABLE_POOL -poolName connection_pool_name [true|false]
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Optional. 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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`-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, the command disables a connection pool named demoPool in a WebLogic domain where the Administration Server is running on a machine named AdminHost and listening at port 7001. This command freezes connections to be enabled later:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic DISABLE_POOL -poolName demoPool 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 connection_pool_name
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Optional. Specify the listen address and listen port of a WebLogic Server instance on which the connection pool has been deployed and is disabled. 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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`-poolName`	Name of the connection pool.

Example

In the following command, a connection pool named demoPool is re-enabled after being disabled:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic ENABLE_POOL -poolName demoPool
```

TEST_POOL

Reserves and releases a connection from the connection pool. The command also runs a test query using the connection, either before reserving the connection or after releasing the connection, to make sure the database is available. This command requires that either testConnsOnReserve or testConnsOnRelease is set to true and testTableName is specified.

Note: The TEST_POOL command tests an individual instance of the connection pool. To test all instances (deployments) of the connection pool, repeat the command for each instance in your configuration.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   TEST_POOL -poolName connection_pool_name
```

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. Providing the url of a server on which the connection pool has NOT been deployed will return incorrect results. If the pool is deployed on multiple servers, run the command multiple times, each time pointing to one server instance on which the connection pool has been deployed. Running the command on only one server instance DOES NOT return the overall, aggregated status of the pool. See Note on page 59. 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 2-3 on page 11 and "Protocol Support" on page 2-13.
`-poolName`	Name of the connection pool as listed in the configuration file (`config.xml`).

Example

This command tests the connection pool registered as `MedRecPool` and deployed on a server that listens on port 8001 of the host ManagedHost:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic -password
weblogic TEST_POOL -poolName MedRecPool
```

If the command succeeds, it returns the following:

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

RESET_POOL

This command closes and reopens the database connections in a connection pool.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   RESET_POOL -poolName connection_pool_name
```

Argument	Definition
`-url [protocol://]listen-address:listen-port`	Optional. 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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`-poolName`	Name of the connection pool as listed in the configuration file (`config.xml`).

Example

This command closes and reopens database connection in the connection pool named demoPool for the WebLogic Server instance listening on port 7001 of the host AdminHost.

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
  -password weblogic RESET_POOL -poolName demoPool
```

EXISTS_POOL

Tests whether a connection pool with a specified name exists in the WebLogic Server domain. 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 [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   EXISTS_POOL -poolName connection_pool_name
```

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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`-poolName`	Name of connection pool.

Example

The following command determines wether or not a pool with a specific name exists:

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
   -password weblogic EXISTS_POOL -poolName demoPool
```

Commands for Managing WebLogic Server MBeans

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

- "Specifying MBean Types" on page 2-62
- "MBean Management Commands" on page 2-62

"WebLogic Server MBean Reference" provides a detailed reference for all 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, see the Type-Safe Access for WebLogic Server MBeans (Deprecated) 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 2-7 is an overview of the MBean management commands.

Table 2-7 MBean Management Command Overview

Command	Description
`CREATE`	Creates an Administration MBean instance. This command cannot be used for Runtime MBeans or Local Configuration MBeans. See "CREATE" on page 2-63.
`DELETE`	Deletes an MBean instance. See "DELETE" on page 2-65.
`GET`	Displays properties of MBeans. See "GET" on page 2-67.
`INVOKE`	Invokes management operations that an MBean exposes for its underlying resource. See "INVOKE" on page 2-69.
`QUERY`	Searches for MBeans whose `WebLogicObjectName` matches a pattern that you specify. See "QUERY" on page 2-71.
`SET`	Sets the specified property values for the named MBean instance. This command cannot be used for Runtime MBeans. See "SET" on page 2-74.

CREATE

Creates an instance of a WebLogic Server Administration MBean. This command cannot be used for Runtime MBeans or Local Configuration MBeans.

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

When you use this command, 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, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities with JMX*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     -url URL
     [ User Credentials Arguments ]
   CREATE -name name -type mbeanType
```

or

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`-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, see "Specifying MBean Types" on page 2-62.
`-mbean objectName`	Fully qualified object name of an MBean in the `WebLogicObjectName` format. For example: `"domain:Type=type,Name=name"` For more information, see 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, see "Example Environment" on page 2-14.

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 [ SSL Arguments ] weblogic.Admin
     [ {-url URL} |
       {-adminurl[protocol://]Admin-Server-listen-address:listen-port}
     ]
     [ User Credentials Arguments ]
   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, see the -url and -adminurl entries in Table 2-3 and "Protocol Support" on page 2-13.
-type *mbeanType*	Deletes all MBeans of the specified type. For more information, see "Specifying MBean Types" on page 2-62.
-mbean *objectName*	Fully qualified object name of an MBean in the WebLogicObjectName format. For example: "*domain*:Type=*type*,Name=*name*" For more information, see the Javadoc for WebLogicObjectName.

Example

The following example deletes the JDBCConnectionPool Administration MBean named myPool:

```
java weblogic.Admin -url AdminHost:7011 -username weblogic
 -password weblogic DELETE -mbean
 MedRec:Name=myPool,Type=JDBCConnectionPool
```

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

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

```
Ok
```

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}. . .}
. . .
```

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 [ SSL Arguments ] weblogic.Admin
     [ {-url URL} |
       {-adminurl[protocol://]Admin-Server-listen-address:listen-port}
     ]
     [ User Credentials Arguments ]
   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, see the -url and -adminurl entries in Table 2-3 and "Protocol Support" on page 2-13.
-type *mbeanType*	Returns information for all MBeans of the specified type. For more information, see "Specifying MBean Types" on page 2-62.
-mbean *objectName*	Fully qualified object name of an MBean in the WebLogicObjectName format: "*domain*:Type=*type*,Location=*location*,Name=*name*" For more information, see 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:7011 -username weblogic
 -password weblogic GET -pretty -mbean
 MedRec:Name=MedRecPool,Type=JDBCConnectionPool
```

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

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 [ SSL Arguments ] weblogic.Admin
     [ {-url URL} |
       {-adminurl[protocol://]Admin-Server-listen-address:listen-port}
     ]
     [ User Credentials Arguments ]
   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, see "Specifying MBean Types" on page 2-62.
-mbean *objectName*	Fully qualified object name of an MBean, in the WebLogicObjectName format: "*domain*:Type=*type*,Location=*location*,Name=*name*" For more information see 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:7011 -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,ServerRuntime=Med
RecServer,Type=JDBCConnectionPoolRuntime"}

Ok
```

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

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, See "WebLogic Server MBean Object Names" in *Developing Custom Management Utilities with JMX*.

The output of the command is as follows:

```
{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 [ SSL Arguments ] weblogic.Admin
     [ {-url URL} |
       {-adminurl[protocol://]Admin-Server-listen-address:listen-port}
     ]
     [ User Credentials Arguments ]
   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, see the -url and -adminurl entries in Table 2-3 and "Protocol Support" on page 2-13.

Argument	Definition
`-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, See "WebLogic Server MBean Object Names" in *Developing Custom Management Utilities with JMX*. (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, see "Example Environment" on page 2-14.

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:7011 -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 or Local Configuration MBeans.

If the command is successful, it returns `OK` and saves the new values to the `config.xml` file.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`-type mbeanType`	Sets the properties for all MBeans of a specific type. For more information, see "Specifying MBean Types" on page 2-62.
`-mbean objectName`	Fully qualified object name of an MBean in the `WebLogicObjectName` format. For example: `"domain:Type=type,Name=name"` For more information, see the Javadoc for `WebLogicObjectName`.

Argument	Definition
`-property property`	The name of the property to be set.
property _value	The value to be set. • Some properties require you to specify the name of a WebLogic Server MBean. In this case, specify the fully qualified object name of an MBean in the `WebLogicObjectName` format. For example: `"domain:Type=type,Name=name"` For more information, see the Javadoc for `WebLogicObjectName`. • 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 `ServerMBean` for a server named `MedRecManagedServer`:

```
java weblogic.Admin -url http://AdminHost:7011
  -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, see "Example Environment" on page 2-14.

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
```

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, see the previous section, "Running Commands in Batch Mode" on page 2-77.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [ {-url URL} |
       {-adminurl[protocol://]Admin-Server-listen-address:listen-port}
     ]
     [ User Credentials Arguments ]
    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, see the `-url` and `-adminurl` entries in Table 2-2 and "Protocol Support" on page 2-13.
`-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. **Note:** Quoted MBean names are allowed in the batch file.

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:7011 -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=7011}}
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, see "Example Environment" on page 2-14.

Commands for Working with Clusters

Table 2-8 is an overview of the commands for working with clusters. Subsequent sections describe command syntax and arguments, and provide an example for each command.

Table 2-8 MBean Management Command Overview

Command	Description
CLUSTERSTATE	Returns the number and state of servers in a cluster. See "CLUSTERSTATE" on page 2-80.
MIGRATE	Migrates a JMS service or a JTA service from one server instance to another within a cluster. See "MIGRATE" on page 2-81.
STARTCLUSTER	Starts all servers in a cluster See "STARTCLUSTER" on page 2-83.
STOPCLUSTER	Forces all servers in a cluster to shut down. See "STOPCLUSTER" on page 2-85.
VALIDATECLUSTERCONFIG	Parses the domain's configuration file and reports any discrepancies in all cluster-related elements. See "VALIDATECLUSTERCONFIG" on page 2-86.

CLUSTERSTATE

Returns the number and state of servers in a cluster.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the -url entry in Table 2-2 and "Protocol Support" on page 2-13.
-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:7011
  -username weblogic -password weblogic
  CLUSTERSTATE -clustername MedRecCluster
```

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

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

```
There are 3 server(s) in cluster: MedRecCluster

The alive servers and their respective states are listed below:
MedRecManagedServer2---RUNNING
MedRecManagedServer3---RUNNING

The other server(s) in the cluster that are not active are:
MedRecManagedServer1
```

MIGRATE

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

For more information about migrating services, see "Service Migration" in the *Using WebLogic Clusters* guide.

Syntax

To migrate JMS resources:

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
  MIGRATE -migratabletarget "serverName (migratable)"
  -destination serverName [-sourcedown] [-destinationdown]
```

To migrate JTA resources:

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
  MIGRATE -jta -migratabletarget 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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`-jta`	Specifies that the migration is a migration of JTA services. If you do not specify this argument, the `MIGRATE` command migrates JMS services.
`-migratabletarget`	Names the server from which the service will migrate. The syntax for the server name varies depending on the type of service you are migrating: • For JMS, specify `"servername (migratable)"` For example, `"myserver (migratable)"` • For JTA, specify *servername* For example, `myserver`

Argument	Definition
-destination	Names the server to which the service 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*.
-destinationdown	Specifies that the destination server is down. When migrating a JMS service to a non-running server instance, the server instance will activate the JMS service upon the next startup. When migrating the JTA Transaction Recovery Service to a non-running server instance, the target server instance 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.

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 must be configured to communicate with a Node Manager. For more information, see "Configure Machines" and "Assign Server Instances to Machines" in the *Administration Console Online Help*.

The Startup Mode field in the Administration Console determines whether a Managed Server starts in the RUNNING, STANDBY, or ADMIN state. See "Specify a Startup Mode" in the *Administration Console Online Help* and "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the -url entry in Table 2-2 and "Protocol Support" on page 2-13.
-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:70011
  -username weblogic -password weblogic
  STARTCLUSTER -clustername Cluster
```

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

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

Forces all servers in a cluster to shut down without waiting for active sessions to complete.

To verify that the command succeeds for a given server instance, refer to the server's local message log and look for the following message:

```
<BEA-000238> <Shutdown has completed.>
```

Review the message time stamp to verify that it was generated by the server session for which you issued the stop command.

For more information about forced shutdowns, see "Forced Shutdown" in *Managing Server Startup and Shutdown*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the `-url` entry in Table 2-2 and "Protocol Support" on page 2-13.
`-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:7011
  -username weblogic -password weblogic
  STOPCLUSTER -clustername MedRecCluster
```

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

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 issued the shutdown request.
Look in the server logs to verify the success or failure of the shutdown
request.
```

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 [ SSL Arguments ] weblogic.Admin
     [-url URL]
     [ User Credentials Arguments ]
   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, see the -url entry in Table 2-2 and "Protocol Support" on page 2-13.
-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:7011
  -username weblogic -password weblogic
  VALIDATECLUSTERCONFIG -configpath
  samples\domains\medrec\config.xml
```

For more information about the environment in which this example runs, see "Example Environment" on page 2-14.

If the cluster configuration contains errors, the command returns a message that describes the error. For example:

```
ERROR:Cluster name:MyCluster has an INVALID Multicast address:null Please
pick an address between (224.0.0.1 and 255.255.255.255)
```

If the cluster configuration is free of errors, the command returns nothing.

CHAPTER 3

Using the WebLogic Server Java Utilities

WebLogic Server provides a number of Java utilities and Ant tasks for performing administrative and programming tasks.

To use these utilities and tasks, you must set your `CLASSPATH` correctly. For more information, see "Modifying the Classpath" on page 4-2.

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

WebLogic Server also provides a number of Ant tasks that automate common application server programming tasks. The Apache Web site provides other useful Ant tasks as well, including tasks for packaging EAR, WAR, and JAR files. For more information, see http://jakarta.apache.org/ant/manual/.

- "appc" on page 3-3
- "AppletArchiver" on page 3-3
- "autotype (deprecated)" on page 3-4
- "BuildXMLGen" on page 3-4
- "CertGen" on page 3-4
- "ClientDeployer" on page 3-7

- "clientgen" on page 3-7
- "Conversion (deprecated)" on page 3-7
- "dbping" on page 3-7
- "DDInit" on page 3-11
- "Deployer" on page 3-12
- "der2pem" on page 3-12
- "ejbc (deprecated)" on page 3-13
- "EJBGen" on page 3-14
- "encrypt" on page 3-14
- "getProperty" on page 3-15
- "host2ior" on page 3-16
- "ImportPrivateKey" on page 3-16
- "jspc (deprecated)" on page 3-19
- "logToZip" on page 3-19
- "MBean Commands" on page 3-20
- "MulticastTest" on page 3-20
- "myip" on page 3-23
- "pem2der" on page 3-23
- "rmic" on page 3-24
- "Schema" on page 3-24
- "showLicenses" on page 3-26
- "source2wsdd (deprecated)" on page 3-26
- "system" on page 3-27
- "ValidateCertChain" on page 3-27
- "verboseToZip" on page 3-28

- "wlappc" on page 3-29
- "wlcompile" on page 3-29
- "wlconfig" on page 3-29
- "wldeploy" on page 3-29
- "wlpackage" on page 3-29
- "wlserver" on page 3-30
- "writeLicense" on page 3-30
- "wsdl2Service" on page 3-32
- "wsdlgen (deprecated)" on page 3-32
- "wspackage (deprecated)" on page 3-33

appc

The appc compiler generates and compiles the classes needed to deploy EJBs and JSPs to WebLogic Server. It also validates the deployment descriptors for compliance with the current specifications at both the individual module level and the application level. See appc Reference in *Programming WebLogic Server Enterprise JavaBeans*.

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.

autotype (deprecated)

Use the `autotype` Ant task to generate non-built-in data type components, such as the serialization class, for Web Services. The fully qualified name for the `autotype` Ant task is `weblogic.ant.taskdefs.webservices.javaschema.JavaSchema`.

Web Services are now a J2EE standard, which has resulted in many changes between 8.1 and 9.0 WebLogic Web Services. For information about what has changed between 8.1 and 9.0 Web Services, and what has been deprecated, see "Differences Between 8.1 and 9.0 WebLogic Web Services" in *Programming Web Services for WebLogic Server.*

For a complete list of Web Services Ant tasks, see "Ant Task Reference" in *Programming Web Services for WebLogic Server.*

BuildXMLGen

Use BuildXMLGen to generate a `build.xml` file for enterprise applications in the split-directory structure. For complete documentation of this utility, see "Building Applications in a Split Development Directory" in *Developing Applications with WebLogic Server.*

CertGen

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

Syntax

```
$ java utils.CertGen

   -certfile <cert_file> -keyfile <private_key_file>
   -keyfilepass <private_key_password>
   [-cacert <ca_cert_file>][-cakey <ca_key_file>]
   [-cakeypass <ca_key_password>]
   [-selfsigned][-strength <key_strength>]
   [-e <email_address>][-cn <common_name>]
   [-ou <org_unit>][-o <organization>]
   [-l <locality>][-s <state>][-c <country_code>]
   [-keyusage [digitalSignature,nonRepudiation,keyEncipherment,
            dataEncipherment,keyAgreement,keyCertSign,
            cRLSign,encipherOnly,decipherOnly]]
```

```
[-keyusagecritical true|false]
[-subjectkeyid <subject_key_identifier>]
[-subjectkeyidformat UTF-8|BASE64]
[-help]
```

Argument	Definition
`-certfile` *cert_file* `-keyfile` *private_key_file*	Respectively, the output file names without extensions for the generated public certificate and private key. The appropriate extensions are appended when the `pem` and `der` files are created.
`-keyfilepass` *private_key_password*	The password for the generated private key.
`-cacert` *ca_cert_file* `-cakey` *ca_key_file* `-cakeypass` *ca_key_password*	Respectively, the public certificate, private key file, and private key password of the CA that will be used as the issuer of the generated certificate. If one or more of these options are not specified, the relevant demonstration CA files will be used: `CertGenCA.der` and `CertGenCAKey.der`. The `CertGen` utility first looks in the current working directory, then in the `WL_HOME/lib` directory.
`-selfsigned`	Generates a self-signed certificate that can be used as a trusted CA certificate. If this argument is specified, the *ca_cert_filename*, *ca_key_filename*, and *ca_key_password* arguments should not be specified.
`-strength` *key_strength*	The length (in bits) of the keys to be generated. The longer the key, the more difficult it is for someone to break the encryption.
`-e` *email_address*	The email address associated with the generated certificate.
`-cn` *common_name*	The name associated with the generated certificate.
`-ou` *org_unit*	The name of the organizational unit associated with the generated certificate.
`-o` *organization*	The name of the organization associated with the generated certificate.

Argument	Definition
`-l` *`locality`*	The name of a city or town.
`-s` *`state`*	The name of the state or province in which the organizational unit (ou) operates if your organization is in the United States or Canada, respectively. Do not abbreviate.
`-c` *`country_code`*	Two-letter ISO code for your country. The code for the United States is US.
`-keyusage [digitalSignature, nonRepudiation,keyEncipherment, dataEncipherment,keyAgreement, keyCertSign,cRLSign, encipherOnly,decipherOnly]`	Generate certificate with a keyusage extension, and with bits set according to the comma-separated list of bit names. Specify a key usage when you want to restrict the operation for a key that could be used for more than one operation.
`-keyusagecritical true|false`	By default, a keyusage extension is marked critical. To generate a certificate with a non-critical extension, use `-keyusagecritical false`.
`-subjectkeyid` *`subject_key_identifier`*	Generates a certificate with the specified subject key identifier.
`-subjectkeyidformat UTF-8|BASE64`	The format of the `subjectkeyid` value; UTF-8 is the default.

Example

By default, the CertGen utility looks for the `CertGenCA.der` and `CertGenCAKey.der` files in the current directory, or in the *`WL_HOME`* directory, as specified in the `weblogic.home` system property or the `CLASSPATH`. Alternatively, you can specify CA files on the command line.

Enter the following command to generate certificate files named `testcert` with private key files named `testkey`:

```
$ java utils.CertGen -keyfilepass mykeypass
-certfile testcert -keyfile testkey
```

```
Generating a certificate with common name return and key strength 1024
issued by CA with certificate from CertGenCA.der file and key from
CertGenCAKey.der file
```

ClientDeployer

You use `weblogic.ClientDeployer` to extract the client-side JAR file from a J2EE EAR file, creating a deployable JAR file. The `weblogic.ClientDeployer` class is executed on the Java command line with the following syntax:

```
java weblogic.ClientDeployer ear-file client
```

The *`ear-file`* argument is an expanded directory (or Java archive file with a `.ear` extension) that contains one or more client application JAR files.

For example:

```
java weblogic.ClientDeployer app.ear myclient
```

where `app.ear` is the EAR file that contains a J2EE client packaged in `myclient.jar`.

Once the client-side JAR file is extracted from the EAR file, use the `weblogic.j2eeclient.Main` utility to bootstrap the client-side application and point it to a WebLogic Server instance as follows:

```
java weblogic.j2eeclient.Main clientjar URL [application args]
```

For example:

```
java weblogic.j2eeclient.Main helloWorld.jar t3://localhost:7001
Greetings
```

clientgen

Use `clientgen` to generate the client-side artifacts, such as the JAX-RPC stubs, needed to invoke a Web Service. See "Ant Task Reference" in *Programming Web Services for WebLogic Server*.

Conversion (deprecated)

WebLogic Server 9.0 does not support conversion or upgrading from a pre-6.0 version of WebLogic Server. To upgrade from version 6.1 or later, see *Upgrading WebLogic Application Environments*.

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 the documentation from your driver vendor. Also see "Using Third-Party Drivers with WebLogic Server" in *Programming WebLogic JDBC*.

Creating a DB2 Package with dbping

With the WebLogic Type 4 JDBC Driver for DB2, you can also use the `dbping` utility to create a package on the DB2 server. When you ping the database with the dbping utility, the driver automatically creates the default package on the database server if it does not already exist. If the default package already exists on the database server, the `dbping` utility uses the existing package.

The default DB2 package includes 200 dynamic sections. You can specify a different number of dynamic sections to create in the DB2 package with the `-d` option. The `-d` option also sets `CreateDefaultPackage=true` and `ReplacePackage=true` on the connection used in the connection test, which forces the DB2 driver to replace the DB2 package on the DB2 server. (See DB2 Connection Properties for more information.) You can use the `-d` option with dynamic sections set at `200` to forcibly recreate a default package on the DB2 server.

Notes: When you specify the `-d` option, the `dbping` utility *recreates* the default package and uses the value you specify for the number of dynamic sections. It does not modify the existing package.

To create a DB2 package, the user that you specify must have CREATE PACKAGE privileges on the database.

Syntax

```
$ java utils.dbping DBMS [-d dynamicSections] user password DB
```

Argument	Definition
DBMS	Varies by DBMS and JDBC driver: DB2B—WebLogic Type 4 JDBC Driver for DB2 JCONN2—Sybase JConnect 5.5 (JDBC 2.0) driver JCONN3—Sybase JConnect 6.0 (JDBC 2.0) driver JCONNECT—Sybase JConnect driver INFORMIXB—WebLogic Type 4 JDBC Driver for Informix MSSQLSERVER4—WebLogic jDriver for Microsoft SQL Server MSSQLSERVERB—WebLogic Type 4 JDBC Driver for Microsoft SQL Server MYSQL— MySQL's Type 4 Driver ORACLE—WebLogic jDriver for Oracle ORACLEB—WebLogic Type 4 JDBC Driver for Oracle ORACLE_THIN—Oracle Thin Driver POINTBASE—PointBase Universal Driver SYBASEB—WebLogic Type 4 JDBC Driver for Sybase
[-d *dynamicSections*]	Specifies the number of dynamic sections to create in the DB2 package. This option is for use with the WebLogic Type 4 JDBC Driver for DB2 only. If the -d option is specified, the driver automatically sets CreateDefaultPackage=true and ReplacePackage=true on the connection and creates a DB2 package with the number of dynamic sections specified.
user	Valid database username for login. Use the same values you use with isql, sqlplus, or other SQL command-line tools. For DB2 with the -d option, the user must have CREATE PACKAGE privileges on the database.

Argument	Definition
password	Valid database password for the user. Use the same values you use with `isql` or `sqlplus`.
DB	Name and location of the database. Use the following format, depending on which JDBC driver you use: `DB2B`—Host:Port/DBName `JCONN2`—Host:Port/DBName `JCONN3`—Host:Port/DBName `JCONNECT`—Host:Port/DBName `INFORMIXB`—Host:Port/DBName/InformixServer `MSSQLSERVER4`—Host:Port/DBName or [DBName@]Host[:Port] `MSSQLSERVERB`—Host:Port/DBName `MYSQL`—Host:Port/DBName `ORACLE`—DBName (as listed in tnsnames.ora) `ORACLEB`—Host:Port/DBName `ORACLE_THIN`—Host:Port/DBName `POINTBASE`—Host[:Port]/DBName `SYBASEB`—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. • *`DBName`* is the name of a database on the DBMS. • *`InformixServer`* is an Informix-specific environment variable that identifies the Informix DBMS server.

Example

```
C:\>java utils.dbping ORACLE_THIN scott tiger dbserver1:1561:demo

**** 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");
  props.put("dll", "ocijdbc9");
  props.put("protocol", "thin");
  java.sql.Driver d =
    Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
  java.sql.Connection conn =
    Driver.connect("jdbc:oracle:thin:@dbserver1:1561:demo", props);
```

ddcreate

This Ant task calls EARInit, which generates an `application.xml` and a `weblogic-application.xml` file for an `EAR`. For more information, see "EarInit" on page 3-12.

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.

In its command-line version, `DDInit` writes new files that overwrite existing descriptor files. If `META-INF` or `WEB-INF` does not exist, `DDInit` creates it.

Specify the type of J2EE deployable unit (either Web Application or Enteprise Application) for which you want deployment descriptors generated by using the `DDInit` command specific to the type, as described below.

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.

```
prompt> 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.

```
prompt> java weblogic.marathon.ddinit.EarInit <module>
```

Deployer

Using the `weblogic.Deployer` tool, you can deploy J2EE applications and components to WebLogic Servers in a command-line or scripting environment. For detailed information on using this tool, see "weblogic.Deployer Command-Line Reference" in *Deploying Applications to WebLogic Server*.

The `weblogic.Deployer` utility replaces the `weblogic.deploy` utility, which has been deprecated.

der2pem

The `der2pem` utility converts an X509 certificate from DER format to PEM format. The `.pem` file is written in the same directory and has the same filename 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
.....................................................................
```

ejbc (deprecated)

See appc Reference in *Programming Weblogic Server Enterprise JavaBeans*.

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.

See EJBGen Reference in *Programming WebLogic Server Enterprise JavaBeans*.

encrypt

The `weblogic.security.Encrypt` utility encrypts cleartext strings for use with WebLogic Server. The utility uses the encryption service of the current directory, or the encryption service for a specified WebLogic Server domain root directory.

Note: An encrypted string must have been encrypted by the encryption service in the WebLogic Server domain where it will be used. If not, the server will not be able to decrypt the string.

You can only run the `weblogic.security.Encrypt` utility on a machine that has at least one server instance in a WebLogic Server domain; it cannot be run from a client.

Note: BEA Systems recommends running the utility from the Administration Server domain directory or on the machine hosting the Administration Server and specifying a domain root directory.

Syntax

```
java   [ -Dweblogic.RootDirectory=dirname ]
       [ -Dweblogic.management.allowPasswordEcho=true ]
       weblogic.security.Encrypt [ password ]
```

Argument	Definition
`weblogic.RootDirectory`	Optional. WebLogic Server domain directory in which the encrypted string will be used. If not specified, the default domain root directory is the current directory (the directory in which the utility is being run).
`weblogic.management.allow PasswordEcho`	Optional. Allows echoing characters entered on the command line. `weblogic.security.Encrypt` expects that no-echo is available; if no-echo is not available, set this property to `true`.

Argument	Definition
password	Optional. Cleartext string to be encrypted. If omitted from the command line, you will be prompted to enter a password.

Examples

The utility returns an encrypted string using the encryption service of the domain located in the current directory.

```
java weblogic.security.Encrypt xxxxxx

{3DES}Rd39isn4LLuF884Ns
```

The utility returns an encrypted string using the encryption service of the specified domain location.

```
java -Dweblogic.RootDirectory=./mydomain weblogic.security.Encrypt xxxxxx

{3DES}hsikci118SKFnnw
```

The utility returns an encrypted string in the current directory, without echoing the password.

```
java weblogic.security.Encrypt

Password:

{3DES}12hsIIn56KKKs3
```

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

-certfile <cert_file> -keyfile <private_key_file>
[-keyfilepass <private_key_password>]
-keystore <keystore> -storepass <storepass> [-storetype <storetype>]
```

```
-alias <alias> [-keypass <keypass>]
[-help]
```

Argument	Definition
cert_file	The name of the certificate associated with the private key.
private_key_file	The name of the generated private key file.
private_key_password	The password for the private key.
keystore	The name of the keystore. A new keystore is created if one does not exist.
storepass	The password to open the keystore.
storetype	The type (format) of the keystore. The *storetype* argument, which is the same as that used by the `keytool` command, specifies the type of Java keystore. The default *storetype* is `jks`, defined by the `keystore.type` property in the `java.security` file: `keystore.type=jks` You can specify another *storetype* (for example, `pcks12` or `nCipher.SWorld`) if a configured security provider supports that type.
alias	The name that is used to look up certificates and keys in the keystore.
keypass	The password of the key entry in the keystore. If *keypass* is not specified, the first default is to look for a *keyfile_pass*, the second default is to look for *storepass*. Note that if you used `CertGen` to create a private keyfile protected by a password (`-keyfilepass` *keyfile_pass*), that password is the one required by `ImportPrivateKey` to extract the key from the keyfile and insert the key in the newly created keystore (which will contain both the certificate(s) from *cert_file* and the private key from *keyfile*).

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

To generate a certificate:

Note: By default, the CertGen utility looks for the `CertGenCA.der` and `CertGenCAKey.der` files in the current directory, or in the *`WL_HOME`*`/server/lib` directory, as specified in the `weblogic.home` system property or the `CLASSPATH`.

Alternatively, you can specify CA files on the command line. If you want to use the default settings, there is no need to specify CA files on the command line.

1. Enter the following command to generate certificate files named `testcert` with private key files named `testkey`:

```
$ java utils.CertGen -keyfilepass mykeypass
-certfile testcert -keyfile testkey

Generating a certificate with common name return and key strength 1024
issued by CA with certificate from CertGenCA.der file and key from
CertGenCAKey.der file
```

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

```
$ java utils.der2pem CertGenCA.der
```

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

```
$ cat testcert.pem CertGenCA.pem >> newcerts.pem
```

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

```
$ java utils.ImportPrivateKey -keystore mykeystore -storepass mypasswd
-keyfile mykey -keyfilepass mykeypass -certfile newcerts.pem -keyfile
testkey.pem -alias passalias


No password was specified for the key entry
Key file password will be used

Imported private key testkey.pem and certificate newcerts.pem
into a new keystore mykeystore of type jks under alias passalias
```

jhtml2jsp

Converts JHTML files to JSP files. Be sure to inspect the results carefully. Given the unpredictability of the JHTML code, `jhtml2jsp` will not necessarily produce flawless translations.

The output is a new JSP file named after the original file.

The HTTP servlets auto-generated from JSP pages differ from the regular HTTP servlets 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 reference these methods to access the `servlet context` or `config` objects, you must 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 must 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 the target directory isn't specified, output is written to the current directory.

jspc (deprecated)

JSP-specific compiler task. Use "appc" on page 3-3.

logToZip

The `logToZip` utility searches an HTTP server log file, 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 shown below, 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
```

MBean Commands

Use the MBean commands (CREATE, DELETE, GET, INVOKE, and SET) to administer MBeans. See "Editing Commands" in *WebLogic Scripting Tool*.

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 the current server.

2. The sequence and sender ID of each message received from any clustered server, including the current 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 and hardware of the WebLogic Server host machine. 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`	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 is 237.0.0.1.)
`-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.)

Argument	Definition
`-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
```

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
.....................................................................
.....................................................................
.....................................................................
.....................................................................
.....................................................................
```

pointbase

PointBase is bundled with WebLogic Server as a sample database. Its documentation is also included at `WL_HOME\common\eval\pointbase\docs`, where `WL_HOME` is the WebLogic Server installation directory, typically `C:\bea\weblogic90`.

The PointBase documentation is also on the PointBase site, at `http://www.pointbase.com/support/docs/overview.html`.

rmic

The WebLogic RMI compiler is a command-line utility for generating and compiling remote objects. Use `weblogic.rmic` to generate dynamic proxies on the client-side for custom remote object interfaces in your application, and to provide hot code generation for server-side objects. See "Using the WebLogic RMI Compiler" in *Programming WebLogic RMI*.

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:

```
$ 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
```

servicegen (deprecated)

The `servicegen` Ant task takes as input an EJB JAR file or a list of Java classes, and creates all the needed Web Service components and packages them into a deployable EAR file.

Web Services are now a J2EE standard, which has resulted in many changes between 8.1 and 9.0 WebLogic Web Services. For information about what has changed between 8.1 and 9.0 Web Services, and what has been deprecated, see "Differences Between 8.1 and 9.0 WebLogic Web Services" in *Programming Web Services for Weblogic Server.*

For a complete list of Web Services Ant tasks, see "Ant Task Reference" in *Programming Web Services for Weblogic Server.*

SearchAndBuild

This Ant task executes `build.xml` files that are included within the `FileSet`. The task assumes that all of the files defined in `FileSet` are valid build files, and executes the `Ant` task of each of them.

Make certain that your `FileSet` filtering is correct. If you include the `build.xml` file that `SearchAndBuildTask` is being called from, you will be stuck in an infinite loop as this task will execute the top level build file—itself—forever. See FileSet at `http://ant.apache.org/manual/CoreTypes/fileset.html`.

Example

```
<project name="all_modules" default="all" basedir=".">
<taskdef name="buildAll"
classname="weblogic.ant.taskdefs.build.SearchAndBuildTask"/>
<target name="all">
<buildAll>
<fileset dir="${basedir}">
<include name="**\build.xml"/>
<exclude name="build.xml"/>
</fileset>
</buildAll>
</target>
</project>
```

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
```

source2wsdd (deprecated)

Generates a `web-services.xml` deployment descriptor file from the Java source file for a Java class-implemented WebLogic Web Service.

Web Services are now a J2EE standard, which has resulted in many changes between 8.1 and 9.0 WebLogic Web Services. For information about what has changed between 8.1 and 9.0 Web

Services, and what has been deprecated, see "Differences Between 8.1 and 9.0 WebLogic Web Services" in *Programming Web Services for Weblogic Server.*

For a complete list of Web Services Ant tasks, see "Ant Task Reference" in *Programming Web Services for Weblogic Server.*

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 * * * * * * *
1.5.0_03
* * * * * * * java.vendor * * * * * * *
BEA Systems, Inc.
* * * * * * * java.class.path * * * * * * *
C:\src_15003jr\bea\weblogic90\server\classes;
C:\dev\src\build\JROCKI~2.0_0\lib\tools.jar;
...
* * * * * * * os.name * * * * * * *
Windows 2000
* * * * * * * os.arch * * * * * * *
x86
* * * * * * * os.version * * * * * * *
5.0
```

ValidateCertChain

WebLogic Server provides the `ValidateCertChain` utility to check whether or not an existing certificate chain will be rejected by WebLogic Server. The utility uses certificate chains from

PEM files, PKCS-12 files, PKCS-12 keystores, and JKS keystores. A complete certificate chain must be used with the utility. The following is the syntax for the `ValidateCertChain` utility:

```
java utils.ValidateCertChain -file pemcertificatefilenamejava
utils.ValidateCertChain -pem pemcertificatefilenamejava
utils.ValidateCertChain -pkcs12store pkcs12storefilenamejava
utils.ValidateCertChain -pkcs12file pkcs12filename passwordjava
utils.ValidateCertChain -jks alias storefilename [storePass]
```

Example of valid certificate chain:

```
java utils.ValidateCertChain -pem zippychain.pemCert[0]:
CN=zippy,OU=FOR
TESTINGONLY,O=MyOrganization,L=MyTown,ST=MyState,C=USCert[1]:
CN=CertGenCAB,OU=FOR
TESTINGONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US

Certificate chain appears valid
```

Example of invalid certificate chain:

```
java utils.ValidateCertChain -jks mykey mykeystoreCert[0]:
CN=corba1,OU=FOR TESTING ONLY,
O=MyOrganization,L=MyTown,ST=MyState,C=US

CA cert not marked with critical BasicConstraint indicating it is a
CACert[1]: CN=CACERT,OU=FOR TESTING ONLY,
O=MyOrganization,L=MyTown,ST=MyState,C=USCertificate chain is invalid
```

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.

Example

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

wlappc

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

For more information, see "Building Modules and Applications Using wlappc" in *Developing Applications with WebLogic Server*.

wlcompile

Use the `wlcompile` Ant task to invoke the `javac` compiler to compile your application's Java files in a split development directory structure. See "Building Applications in a Split Development Directory" in *Developing Applications with WebLogic Server*.

wlconfig

The wlconfig Ant task enables you to configure a WebLogic Server domain by creating, querying, or modifying configuration MBeans on a running Administration Server instance. For complete documentation on this Ant task, see "Using Ant Tasks to Configure a WebLogic Server Domain" in *Developing Applications with WebLogic Server*.

wldeploy

The `wldeploy` Ant task enables you to perform `Deployer` functions using attributes specified in an Ant task. See "Deploying and Packaging from a Split Development Directory" in *Developing Applications with WebLogic Server*.

wlpackage

You use the `wlpackage` Ant task to package your split development directory application as a traditional EAR file that can be deployed to WebLogic Server. See "Deploying and Packaging from a Split Development Directory" in *Developing Applications with WebLogic Server*.

wlserver

The `wlserver` Ant task enables you to start, reboot, shutdown, or connect to a WebLogic Server instance. The server instance may already exist in a configured WebLogic Server domain, or you can create a new single-server domain for development by using the `generateconfig=true` attribute. For complete documentation on this Ant task, see "Starting Servers and Creating Domains Using the wlserver Ant Task" in *Developing Applications with WebLogic Server*.

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 -Dbea.home=path
```

Argument	Definition
`-nowrite`	Optional. Sends the output to `stdout` instead of `writeLicense.txt`.
`-Dbea.home`	Required. Sets WebLogic system home (the root directory of your WebLogic Server installation).

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 2000 Output

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

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

1.5.0_03
* * * * * * java.vendor * * * * * * *

BEA Systems, Inc.
* * * * * * java.class.path * * * * * * *

C:\src_15003jr\bea\weblogic90\server\classes;
C:\dev\src\build\JROCKI~2.0_0\lib\tools.jar;
...
* * * * * * os.name * * * * * * *

Windows 2000
* * * * * * os.arch * * * * * * *

x86
* * * * * * os.version * * * * * * *

5.0
* * * * * 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: 08/22/2005 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
```

wsdl2Service

The `wsdl2Service` Ant task is a Web Services tool that takes as input an existing WSDL file and generates the Java interface that represents the implementation of your Web Service and the `web-services.xml` file that describes the Web Service. See "Iterative Development of WebLogic Web Services Starting From a WSDL File: Main Steps" in *Programming Web Services for Weblogic Server*.

wsdlgen (deprecated)

The `wsdlgen` Ant task is a Web Services tool that generates a WSDL file from the EAR and WAR files that implement your Web Service.

Web Services are now a J2EE standard, which has resulted in many changes between 8.1 and 9.0 WebLogic Web Services. For information about what has changed between 8.1 and 9.0 Web Services, and what has been deprecated, see "Differences Between 8.1 and 9.0 WebLogic Web Services" in *Programming Web Services for Weblogic Server*.

For a complete list of Web Services Ant tasks, see "Ant Task Reference" in *Programming Web Services for Weblogic Server*.

wspackage (deprecated)

Use the Web Services `wspackage` Ant task to package the various components of a WebLogic Web Service into a new deployable EAR file and add extra components to an already existing EAR file.

Web Services are now a J2EE standard, which has resulted in many changes between 8.1 and 9.0 WebLogic Web Services. For information about what has changed between 8.1 and 9.0 Web Services, and what has been deprecated, see "Differences Between 8.1 and 9.0 WebLogic Web Services" in *Programming Web Services for Weblogic Server*.

For a complete list of Web Services Ant tasks, see "Ant Task Reference" in *Programming Web Services for Weblogic Server*.

CHAPTER 4

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 prompt (shell), indirectly through scripts, or through the Node Manager.

This section describes the following:

- "Required Environment and Syntax for weblogic.Server" on page 4-1
- "Default Behavior" on page 4-3
- "weblogic.Server Configuration Options" on page 4-4
- "Using the weblogic.Server Command Line to Start a Server Instance" on page 4-28
- "Using the weblogic.Server Command Line to Create a Domain" on page 4-29
- "Verifying Attribute Values That Are Set on the Command Line" on page 4-31

For information about using scripts to start an instance of WebLogic Server, see "Starting an Administration Server with a Startup Script" and "Starting Managed Servers With a Startup Script" in *Managing Server Startup and Shutdown*.

For information about using the Node Manager to start an instance of WebLogic Server, see "Using Node Manager to Control Servers" in *Managing Server Startup and Shutdown*.

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 *Installation Guide*.
2. If desired, modify the `CLASSPATH` environment variable, as described in "Modifying the Classpath" on page 4-2.
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/platform/suppconfigs/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.

Modifying the Classpath

After installation, WebLogic Server's classpath is already set, but you may choose to modify it for a number of reasons such as adding a patch to WebLogic Server, updating the version of PointBase you are using, or adding support for Log4j logging.

To apply a patch to ALL of your WebLogic Server domains without the need to modify the classpath of a domain, give the patch JAR file the name, `weblogic_sp.jar`, and copy it into the *`WL_HOME`*`/server/lib` directory. The `commEnv.cmd/sh` script will automatically include a JAR named `weblogic_sp` on the classpath for you.

If you would rather not use the name `weblogic_sp.jar` for your patch file or you would just like to make sure a JAR file, such as one mentioned below, comes before `weblogic.jar` on the classpath:

- For ALL domains, edit the `commEnv.cmd/sh` script in *`WL_HOME`*`/common/bin` and prepend your JAR file to the `WEBLOGIC_CLASSPATH` environment variable.
- To apply a patch to a SPECIFIC WebLogic Server domain, edit the `setDomainEnv.cmd/sh` script in that domain's `bin` directory, and prepend the JAR file to the `PRE_CLASSPATH` environment variable.

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

`WL_HOME``/common/eval/pointbase/lib/pbembedded51.jar and pbclient51.jar`

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

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

If you use Log4j logging, include the following file on the classpath:

```
WL_HOME/server/lib/log4j.jar
```

The shell environment in which you run a server determines which character you use to separate path elements. On Windows, you typically use a semicolon (;). In a BASH shell, you typically use a colon (:).

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 4-2, when you enter the command `java weblogic.Server` with no options, WebLogic Server does the following:

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

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

   b. If there are multiple server instances defined in `config/config.xml`:
   – If an Administration Server is defined, it looks for the server with that name.
   – If an Administration Server is not defined, it 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 prompts you to create one. If you respond `y`, WebLogic Server does the following:

   a. Creates a server configuration named `myserver`, and persists the configuration in a file named `config/config.xml`.

      Any options that you specify are persisted 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.

      You can configure WebLogic Server to make backup copies of the configuration files. This facilitates recovery in cases where configuration changes need to be reversed or the unlikely case that configuration files become corrupted. For more information, see "Configuration File Archiving" in *Understanding Domain Configuration*.

   b. Uses the username and password that you supply to create a user with administrative privileges. It stores the definition of this user along with other basic, security-related data in *`domain_name`*`/security` files named `DefaultAuthenticatorInit.ldift`, `DefaultRoleMapperInit.ldift`, and `SerializedSystemIni.dat`.

      WebLogic Server also encrypts and stores your username and password in a *`server_name`*`/security/boot.properties` file, which enables you to bypass the login prompt during subsequent instantiations of the server. For more information, see "Boot Identity Files" in *Managing Server Startup and Shutdown*.

   c. Creates two scripts, `bin/startWebLogic.cmd` and `bin/startWebLogic.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 `startWebLogic` script contains comments that describe each option.

   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.

weblogic.Server Configuration Options

You can use `weblogic.Server` options to configure the attributes of a server instance. The following attributes are commonly used when starting a server instance:

- "JVM Parameters" on page 4-5
- "Location of License and Configuration Data" on page 4-6

WebLogic Server provides other startup options that enable you to temporarily override a server's saved configuration. For information about these startup options, see "Options that Override a Server's Configuration" on page 4-9.

Unless you are creating a new domain as described in "Using the weblogic.Server Command Line to Create a Domain" on page 4-29, 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 WebLogic Scripting Tool (WLST) to modify the `config.xml` file. See "Creating and Configuring WebLogic Domains Using WLST Offline" in *WebLogic Scripting Tool*.

For information on verifying the WebLogic Server attribute values that you set, see "Verifying Attribute Values That Are Set on the Command Line" on page 4-31.

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, see the documentation for your specific JVM. For a list of JVMs that can be used with WebLogic Server, see the Supported Configurations page at http://e-docs.bea.com/platform/suppconfigs/index.html.

Table 4-1 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 the default allocation of 256 megabytes of Java heap memory to the WebLogic Server. To do so, start the server using the `java -Xms256m` and `-Xmx512m` options. 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 "Modifying the Classpath" on page 4-2. Instead of using this argument, you can use the `CLASSPATH` environment variable 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, see the Supported Configurations page at http://e-docs.bea.com/platform/suppconfigs/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 4-2 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. See "A Server's Root Directory" in *Understanding Domain Configuration*. By default, the root directory is the directory from which you issue the start command.

`-Dweblogic.ConfigFile=`*`file_name`*	**Note:** This option was removed as of WebLogic Server 9.0.
	Specifies a configuration file for your domain. The *`file_name`* value must see a valid XML file that conforms to the schema as defined in the "BEA WebLogic Server Configuration Reference".
	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`
	If you do not specify this value, the default is `config/config.xml` in the server's root directory.

Table 4-2 Options for Indicating the Location of License and Configuration Data

Option	Description
`-Dweblogic.management.GenerateDefaultConfig=true`	Prevents the `weblogic.Server` class from prompting for confirmation when creating a `config.xml` file. Valid only if you invoke `weblogic.Server` in an empty directory. See "Default Behavior" on page 4-3.
`-Dweblogic.Domain=`*`domain`*	Specifies the name of the domain. If you are using `weblogic.Server` to create a domain, you can use this option to give the domain a specific name. In addition, this option supports a directory structure that WebLogic Server required in releases prior to 7.0 and continues to support in current releases. Prior to 7.0, all configuration files were required to be located at the following pathname: `.../config/`*`domain_name`*`/config.xml` where *`domain_name`* is the name of the domain. If your domain's configuration file conforms to that pathname, and if you invoke the `weblogic.Server` command from a directory other than `config/`*`domain_name`*, you can include the `-Dweblogic.Domain=`*`domain`* argument to cause WebLogic Server to search for a `config.xml` file in a pathname that matches `config/`*`domain_name`*`/config.xml`.

For information on how a Managed Server retrieves its configuration data, see the `-Dweblogic.management.server` entry in Table 4-3.

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, see "Verifying Attribute Values That Are Set on the Command Line" on page 4-31.

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` (where *`WL_HOME`* is the directory in which you installed WebLogic Server):

```
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=http://localhost:7001
-Dweblogic.RootDirectory=c:\my_domains\SimpleDomain weblogic.Server
```

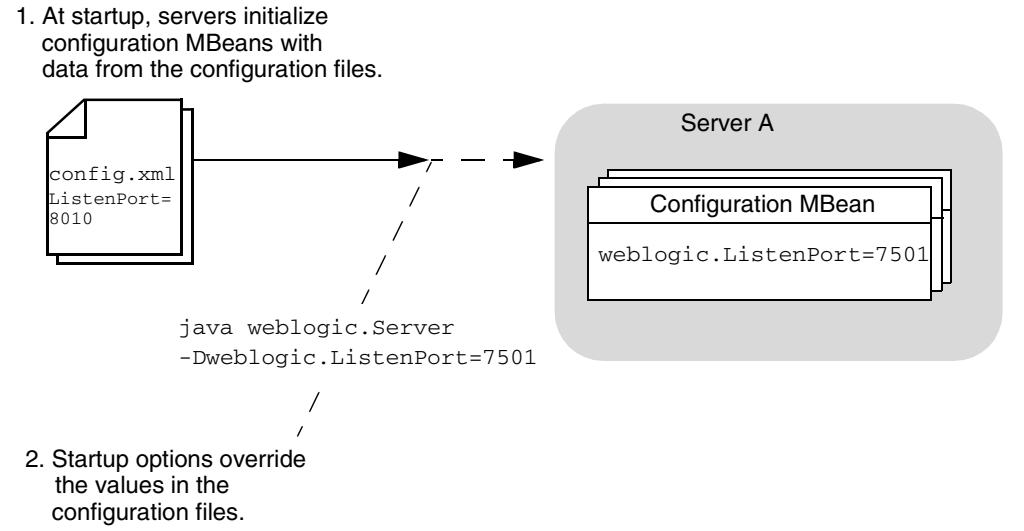
Options that Override a Server's Configuration

In most cases, you do not use startup options to override the configuration that is saved in the domain's `config.xml` file. However, in some extraordinary cases you might need to do so.

Caution: When you use a startup option to override a configuration value, the server instance uses this value for the duration of its life cycle. Even if you use the Administration Console, the WebLogic Scripting Tool, or some other utility to change the value in the configuration, the value will remain overridden until you restart the server without using the override.

For example, in a production environment, your organization might have a policy against modifying the domain's `config.xml` file, but you need to shut down the Administration Server and restart it using a temporary listen port. In this case, when you use the `weblogic.Server` command to start the Administration Server, you can include the `-Dweblogic.ListenPort=7501` startup option to change the listen port for the current server session. The server instance initializes its configuration MBeans from the `config.xml` file but substitutes `7501` as the value of its listen port. When you subsequently restart the server without passing the startup option, it will revert to using the value from the `config.xml` file, `8010`. (See Figure 4-1.)

Figure 4-1 Overriding config.xml Values



The following options temporarily override a server's configuration:

- "Server Communication" on page 4-10
- "SSL" on page 4-15
- "Security" on page 4-18
- "Message Output and Logging" on page 4-24
- "Other Server Configuration Options" on page 4-25
- "Clusters" on page 4-28

Server Communication

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

Table 4-3 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-wide administration port, `port` must specify the domain-wide administration port. For more information on configuring a connection to the Administration Server, see "Configuring a Connection to the Administration Server" in *Managing Server Startup and Shutdown*.

Table 4-3 Options for Configuring Server Communication (Continued)

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 WLST to modify the `config.xml` file. For more information, see "Configure listen addresses" in the *Administration Console Online Help* and "Creating and Configuring WebLogic Domains Using WLST Offline" in *WebLogic Scripting Tool*.
`-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 WLST to modify the `config.xml` file. The default listen port is 7001. For more information, see "Configure listen ports" in the *Administration Console Online Help* and "Creating and Configuring WebLogic Domains Using WLST Offline" in *WebLogic Scripting Tool*.

Table 4-3 Options for Configuring Server Communication (Continued)

Option	Description
`-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 WLST to modify the `config.xml` file. The default SSL listen port is 7002. For more information, see "Configure listen ports" in the *Administration Console Online Help* and "Creating and Configuring WebLogic Domains Using WLST Offline" in *WebLogic Scripting Tool*.
`-Dweblogic.management.discover={true | false}`	**Note:** This option was removed as of WebLogic Server 9.0. 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 communicate with all known Managed Servers and inform them that the Administration Server is running. A false value prevents an Administration Server 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. In WebLogic Server 9.0, this command is deprecated because if an Administration Server stops running while the Managed Servers in the domain continue to run, each Managed Server will periodically attempt to reconnect to the Administration Server at the interval specified by the `ServerMBean` attribute `AdminReconnectIntervalSecs`. For more information, see "Managed Servers and Re-started Administration Server" in *Managing Server Startup and Shutdown*.

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, see "Verifying Attribute Values That Are Set on the Command Line" on page 4-31.

SSL

Each Weblogic Server instance uses an instance of `weblogic.management.configuration.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 4-4 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 WLST to modify the `config.xml` file. For more information, see "Using Hostname Verification" in *Securing WebLogic Server*.
`-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.

Table 4-4 Options for Configuring SSL

Option	Description
`-Dweblogic.security.SSL.sessionCache.ttl=`*`sessionCacheTimeToLive`*	Modifies the default server-session time-to-live for SSL session caching. The *`sessionCacheTimeToLive`* value specifies (in milliseconds) the time to live for the SSL session. The default value is 90000 milliseconds (90 seconds). This means if a client accesses the server again (via the same session ID) within 90 seconds, WebLogic Server will use the existing SSL session. You can change this value by setting -Dweblogic.security.SSL.sessionCache.ttl in the server startup script. For `sessionCache.ttl`: • The minimum value is `1` • The maximum value is `Integer.MAX_VALUE` • The default value is `90000`
`-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`. BEA recommends that you do not use the demonstration certificate authorities in any type of production deployment. For more information, see "Configuring SSL" in the *Securing 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, see “Verifying Attribute Values That Are Set on the Command Line” on page 4-31.

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, see "Verifying Attribute Values That Are Set on the Command Line" on page 4-31.

Security

The following table describes the options for configuring general security parameters.

Table 4-5 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, see "Users, Groups, an Security Roles" in *Securing WebLogic Resources*. This option prevents a server instance from using any boot identity file and overrides other startup options that cause a server to use boot identity files. For more information, see "Boot Identity Files" in *Managing Server Startup and Shutdown*.
`-Dweblogic.management.password=`*`password`*	Specifies the user password. This option prevents a server instance from using any boot identity file and overrides other startup options that cause a server to use boot identity files. For more information, see "Boot Identity Files" in *Managing Server Startup and Shutdown*. **Note:** If you supply the password, but no username, you will be prompted for both the username and the password.
`-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 was used to start the server. Do not specify this argument in a server's `ServerStartMBean`. For more information, see "Specifying User Credentials When Starting a Server with the Node Manager" in *Managing Server Startup and Shutdown*. BEA recommends 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, see "Boot Identity Files" in *Managing Server Startup and Shutdown*.

Table 4-5 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=`*`WL_HOME`*`\mydomain\servers\myserver\security\boot.properties` If you do not specify a filename, a server instance or the `weblogic.Admin SHUTDOWN` and `FORCESHUTDOWN` commands use the `boot.properties` file in the server's root directory. If there is no boot identity file: • When starting a server, the server instance prompts you to enter a username and password. • When using the `weblogic.Admin SHUTDOWN` and `FORCESHUTDOWN` commands, you must use the `-username` and `-password` arguments to provide user credentials. **Note:** The weblogic.Admin utility is deprecated in WebLogic Server 9.0. BEA Systems recommends that you use the WebLogic Scripting Tool (WLST) for equivalent functionality such as `SHUTDOWN` and `FORCESHUTDOWN`. For more information on using these commands, see "Life Cycle Commands" in the *WLST Command and Variable Reference*.
`-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, see "Users, Groups, an Security Roles" in *Securing WebLogic Resources*.

Table 4-5 Options for General Security Parameters

Option	Description
`-Djava.security.manager` `-Djava.security.policy[=]=`*`filename`*	Standard J2EE options that enable the Java security manager and specify 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`. Using `-Djava.security.policy==`*`filename`* (note the double equal sign (==)) causes the policy file to override any default security policy. This causes WebLogic Server to ignore any policy files that are used for servlet and EJB authorization when JACC is enabled. A single equal sign (=) causes the policy file to be appended to an existing security policy. For more information, see "Using the Java Security Manager to Protect WebLogic Resources" in the *Programming WebLogic Security* guide.
`-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 option when starting WebLogic Server to override this problem. This startup option 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.

Table 4-5 Options for General Security Parameters

Option	Description
`-Dweblogic.management.anonymousAdminLookupEnabled=true`	Enables you to retrieve an `MBeanHome` interface without specifying user credentials. The `MBeanHome` interface is part of the WebLogic Server JMX API. If you retrieve `MBeanHome` without specifying user credentials, the interface gives you read-only access to the value of any MBean attribute that is not explicitly marked as protected by the Weblogic Server MBean authorization process. This startup option overrides the Anonymous Admin Lookup Enabled setting on the *`domain_name`*→Security→General page in the Administration Console. By default, the `MBeanHome` API allows access to MBeans only for WebLogic users who are in one of the default security roles. For more information, see "Users, Groups, an Security Roles" in *Securing WebLogic Resources*.

Table 4-5 Options for General Security Parameters

Option	Description
`-Dweblogic.security.identityAssertionTTL=`*`seconds`*	Configures the number of seconds that the Identity Assertion cache stores a Subject. When using an Identity Assertion provider (either for an X.509 certificate or some other type of token), Subjects are cached within the server. This greatly enhances performance for servlets and EJB methods with `<run-as>` tags as well as for other places where identity assertion is used but not cached (for example, signing and encrypting XML documents). There might be some cases where this caching violates the desired semantics. By default, Subjects remain in the cache for 300 seconds, which is also the maximum allowed value. Setting the value to `-1` disables the cache. Setting a high value generally improves the performance of identity assertion, but makes the Identity Assertion provider less responsive to changes in the configured Authentication provider. For example, a change in the user's group will not be reflected until the Subject is flushed from the cache and recreated.

`-Djava.security.manager`
 	
`-Djava.security.policy=<insert the location of your policy file here>`
 	
`-Djavax.security.jacc.PolicyConfigurationFactory.provider=weblogic.security.jacc.simpleprovider.PolicyConfigurationFactoryImpl`
 	
`-Djavax.security.jacc.policy.provider=weblogic.security.jacc.simpleprovider.SimpleJACCPolicy`
 	
`-Dweblogic.security.jacc.RoleMapperFactory.provider=weblogic.security.jacc.simpleprovider.RoleMapperFactoryImpl`
	Defining these five system properties is required to enable the use of the JACC provider in the security realm. When these providers are in use, the JACC handles authorization decisions for the EJB and Servlet containers for external applications. Any other authorization decisions for internal applications are handled by the authorization in the WebLogic Security framework. JACC authorization requires the use of J2SE security and therefore requires that WebLogic Server be booted with a J2EE security manager and a policy file (specified by the server startup properties, `java.security.manager` and `java.security.policy`). For more information, see "Using the Java Security Manager to Protect WebLogic Resources" in *Programming WebLogic Security*.
	
	The WebLogic JACC implementation expects that the policy object is the default `sun.security.provider.PolicyFile` class.
	
	When starting, WebLogic Server attempts to locate and instantiate the classes specified by the JACC startup properties and fails if it cannot find or instantiate them (if, for example, the files specified by the startup properties are not valid classes).

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, see "Verifying Attribute Values That Are Set on the Command Line" on page 4-31.

Message Output and Logging

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

Table 4-6 Options for Configuring Message Output

Option	Description
`-Dweblogic.Stdout="filename"`	Redirects the server and 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. For more information, see "Redirect JVM output" in the *Administration Console Online Help*.
`-Dweblogic.Stderr="filename"`	Redirects the server and 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. For more information, see "Redirecting JVM output" in *Configuring Log Files and Filtering Log Messages*.
`-Dweblogic.AdministrationMBeanAuditingEnabled={true | false}`	Determines whether the Administration Server emits configuration auditing log messages when a user changes the configuration or invokes management operations on any resource within a domain. By default, the Administration Server does not emit configuration auditing messages. See "Enable configuration auditing" 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, see "Verifying Attribute Values That Are Set on the Command Line" on page 4-31.

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, see "Verifying Attribute Values That Are Set on the Command Line" on page 4-31.

Other Server Configuration Options

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

Table 4-7 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}`	This attribute is **deprecated** in WebLogic Server 9.0. 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`*`/autodeploy` directory. If you do not specify this option, the assumed value is `false`. To enable production mode, you can use WLST to set `DomainMBean.isProductionModeEnabled` to `true`, or use the Administration Console. See "Change to production mode" in the *Administration Console Online Help*.
`-Dweblogic.management.startupMode=`*`STARTUPMODE`*	• `STANDBY` starts a server and places it in the `STANDBY` state. See "`STANDBY` state" in *Managing Server Startup and Shutdown*. To use this startup argument, the domain must be configured to use the domain-wide administration port. For information about administration ports, see "Administration Port and Administrative Channel" in *Configuring WebLogic Server Environments* and "Configure the domain-wide administration port" in the *Administration Console Online Help*. • `ADMIN` starts a server and places it in the `ADMIN` state. See "`ADMIN` state" in *Managing Server Startup and Shutdown*. Specifying the startup mode 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 WLST 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.

Table 4-7 Options for Configuring Server Attributes

Option	Description
`-Dweblogic.apache.xerces.maxentityrefs=`*`numerical-value`*	Limits the number of entities in an XML document that the WebLogic XML parser resolves. If you do not specify this option, the XML parser that WebLogic Server installs resolves 10,000 entity references in an XML document, regardless of how many an XML document contains.
`-Dweblogic.jsp.windows.caseSensitive=true`	Causes the JSP compiler on Windows systems to preserve case when it creates output files names. See "Using the WebLogic JSP Compiler" in *Developing Web Applications, Servlets, and JSPs for WebLogic Server*.
`-Dweblogic.servlet.optimisticSerialization=true`	When `optimistic-serialization` is turned on, WebLogic Server does not serialize-deserialize context and request attributes upon `getAttribute(name)` when the request is dispatched across servlet contexts. This means that you must make sure that the attributes common to Web applications are scoped to a common parent classloader (application scoped) or you must place them in the system classpath if the two Web applications do not belong to the same application. When `optimistic-serialization` is turned off (default value), WebLogic Server serialize-deserializes context and request attributes upon `getAttribute(name)` to avoid the possibility of ClassCastExceptions. The optimistic-serialization value can also be specified at domain level in the `WebAppContainerMBean`, which applies for all Web applications. The value in weblogic.xml, if specified, overrides the domain level value. The default value is false.

The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, see "Verifying Attribute Values That Are Set on the Command Line" on page 4-31.

Clusters

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

Table 4-8 Options for Configuring Cluster Attributes

Option	Description
`-Dweblogic.cluster.multicastAddress`	Determines the Multicast Address that clustered servers use to send and receive cluster-related communications. By default, a clustered server refers to the Multicast Address that is defined in the `config.xml` file. Use this option to override the value in `config.xml`. **Note:** The Administration Console does not display values that you set on the command line. For information on verifying the attribute values that you set, see "Verifying Attribute Values That Are Set on the Command Line" on page 4-31. Regardless of how you set the Multicast Address, all servers in a cluster must communicate at the same Multicast Address.

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 root of the domain directory, usually *BEA_HOME*`\user_projects\domains\`*DOMAIN_NAME*. For example, change to the *WL_HOME*`\samples\domains\medrec` directory.

3. To start an Administration Server, enter the following command:
   `java weblogic.Server`

   Note: The password you use must be a string of at least 8 case-sensitive characters. The space character is not supported. For more information, see "Configure an

Administrator Username and Password" in *Creating WebLogic Domains Using the Configuration Wizard*.

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:7011
   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 4-3, 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=7001
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 then 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. For more information about setting the listen address, see "Configure the listen addresses" in the *Administration Console Online Help*.
- The Administration Server listens on port 7001.

Entering the `weblogic.Server` command as described in this section creates the following files:

- `config.xml`
- `DefaultAuthenticatorInit.ldift`, `DefaultRoleMapperInit.ldift`, and `SerializedSystemIni.dat`, which store basic security-related data.
- `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, see "Boot Identity Files" in *Managing Server Startup and Shutdown*.
- `startWebLogic.cmd` and `startWebLogic.sh`, that you can use to start subsequent instantiations of the server.

Note: Invoking `weblogic.Server` in an empty directory results in implicit domain creation which uses the same configuration process as WLST offline and the Configuration Wizard and thus ensures that you always see uniform domains. As a result, implicitly creating a domain in an empty directory using `weblogic.Server` may take around 15 seconds.

Verifying Attribute Values That Are Set on the Command Line

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. To see the values that are in a server's local configuration MBean, use WLST as follows:

1. Follow "Main Steps for Using WLST" which includes "Setting Up Your Environment" and "Invoking WLST" in *WebLogic Scripting Tool*.

   ```
   >java weblogic.WLST
   ```

2. Start a WebLogic Server instance (see Starting and Stopping Servers) and connect WLST to the server using the `connect` command. For detailed information about the `connect` command, see "connect" in the *WLST Command and Variable Reference*.

   ```
   wls:/(offline)> connect('username','password','t3s://localhost:7002')
   Connecting to weblogic server instance running at t3s://localhost:7002
   as username weblogic ...

   wls:/mydomain/serverConfig>
   ```

3. For example, to determine the multicast address that a cluster member is using, connect WLST to that server instance and enter the following commands:

   ```
   wls:/mydomain/serverConfig> cd('Clusters/cluster_name')
   wls:/mydomain/serverConfig/Clusters/mycluster>
   cmo.getMulticastAddress()

   '239.192.0.0'
   ```

4. To determine the severity level of messages that the server instance prints to standard out, connect WLST to that server instance and enter the following commands:

   ```
   wls:/mydomain/serverConfig> cd('Servers/server_name/Log/server_name')
   wls:/mydomain/serverConfig/Servers/myserver/Log/myserver>cmo.getStdoutS
   everity()

   'Notice'
   ```

For more information on using WLST, see *WebLogic Scripting Tool*. For more information about configuration MBeans, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities with JMX*.

CHAPTER 5

WebLogic SNMP Agent Command-Line Reference

WebLogic Server can use Simple Network Management Protocol (SNMP) to communicate with enterprise-wide management systems. The WebLogic Server subsystem that gathers WebLogic management data (managed objects), converts it to SNMP communication modules (trap notifications), and forwards the trap notifications to third-party SNMP management systems is called the WebLogic SNMP agent. The WebLogic SNMP agent runs on the Administration Server and collects managed objects from all Managed Servers within a domain.

The WebLogic SNMP agent provides a command-line interface that lets you:

- Retrieve WebLogic Server managed objects.
- Generate and receive WebLogic Server traps for testing purposes.

The following sections describe working with the WebLogic SNMP agent through its command-line interface:

- "Required Environment for the SNMP Command-Line Interface" on page 5-2
- "Syntax and Common Arguments for the SNMP Command-Line Interface" on page 5-2
- "Commands for Retrieving WebLogic Server Managed Objects" on page 5-4
- "Commands for Testing Traps" on page 5-9

For more information about using SNMP with WebLogic Server, see:

- WebLogic SNMP Management Guide
- WebLogic Server SNMP MIB Reference

Required Environment for the SNMP Command-Line Interface

To set up your environment for the WebLogic SNMP agent command-line interface:

1. Install and configure the WebLogic Server software, as described in the *Installation Guide*.
2. If you want to retrieve WebLogic Server managed objects, enable the WebLogic SNMP agent as described in "Use SNMP to Monitor WebLogic Server" in the *Administration Console Online Help*.
3. Open a command prompt (shell) and invoke the following script:

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

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

   The script adds a supported JDK to the shell's `PATH` environment variable and adds WebLogic Server classes to the `CLASSPATH` variable.

Syntax and Common Arguments for the SNMP Command-Line Interface

All WebLogic SNMP agent commands take the following form:

`java` *`command-name arguments`*

Table 5-1 describes arguments that are common to most WebLogic SNMP agent commands.

Table 5-1 Common Command Line Arguments

Argument	Definition
`-d`	Includes debugging information and packet dumps in the command output.
`-v {v1 | v2}`	Specifies whether to use SNMPv1 or SNMPv2 to communicate with the SNMP agent. You must specify the same SNMP version that you set in the Trap Version field when you configured the SNMP agent (as described in "Configure the SNMP Agent" in the *Administration Console Online Help*) If you do not specify a value, the command assumes `-v v1`.

Table 5-1 Common Command Line Arguments

Argument	Definition
`-c snmpCommunity [@server_name| @domain_name`	The community name that you set for the WebLogic SNMP agent **and** optionally specifies the server instance that hosts the objects with which you want to interact. To request a managed object on the Administration Server, specify: *snmpCommunity* where *snmpCommunity* is the SNMP community name that you set in the Community Prefix field when you configured the SNMP agent (as described in "Configure the SNMP Agent" in the *Administration Console Online Help*). To request a managed object on a single Managed Server, specify: *snmpCommunity@server_name* where *server_name* is the name of the Managed Server. To request a managed object for all server instances in a domain, specify a community string with the following form: *snmpCommunity@domain_name* where *domain_name* is the name of the WebLogic Server domain. If you do not specify a value for this argument, the command assumes `-c public`, which uses the default community name, and assumes that the specified managed object is on the Administration Server.
`-p snmpPort`	The port number on which the WebLogic SNMP agent listens for requests. If you do not specify a value, the command assumes `-p 161`.
`-t timeout`	The number of milliseconds the command waits to successfully connect to the SNMP agent. If you do not specify a value, the command assumes `-t 5000`.
`-r retries`	The number of times the command retries unsuccessful attempts to connect to the SNMP agent. If you do not specify a value, the command exits on the first unsuccessful attempt.
host	The DNS name or IP address of the computer that hosts the WebLogic Server Administration Server, which is where the WebLogic SNMP agent runs.

Commands for Retrieving WebLogic Server Managed Objects

Table 5-2 is an overview of commands that retrieve WebLogic Server managed objects and object instances.

Table 5-2 Overview of Commands for Retrieving Data from WebLogic Server Managed Objects

Command	Description
`snmpwalk`	Returns all managed objects and instances that are below a specified node in the MIB. See "snmpwalk" on page 5-4.
`snmpgetnext`	Returns the managed object or instance that immediately follows an OID that you specify. See "snmpgetnext" on page 5-6.
`snmpget`	Returns managed object instances that correspond to one or more OIDs. See "snmpget" on page 5-8.

snmpwalk

Returns all managed objects or instances that are below a specified node in the MIB.

If you specify the OID for a tabular object, the command returns all of its object instances along with all related (child) objects and instances.

Syntax

```
java snmpwalk [-d] [-v (v1,v2)] [-c snmpCommunity] [-p snmpPort]
              [-t timeout] [-r retries] host OID
```

Argument	Definition
OID	The object ID of the node from which you want to retrieve a set of child objects and instances. Start the value with '.'; otherwise, references are assumed to be relative to the standard MIB (`.1.3.6.1.2.1`), not the WebLogic Server MIB.

Example

The following example retrieves the names of all applications that have been deployed on the Administration Server. The managed object for an application name is `applicationRuntimeName`, which is a child of the `applicationRuntimeTable` object. (See WebLogic Server SNMP MIB Reference.)

```
java snmpwalk localhost .1.3.6.1.4.1.140.625.105.1.15
```

If you invoke this command from a computer that is running the example MedRecServer, the command returns output similar to the following truncated output. Note that the output includes the full OID for each instance of the `applicationRuntimeName` object.

```
Object ID:
.1.3.6.1.4.1.140.625.105.1.15.32.49.102.98.97.100.97.102.99.57.48.50.
102.48.98.53.54.100.100.49.54.50.54.99.54.99.49.97.97.98.53.100.97
STRING: MedRecServer_uddiexplorer

Object ID:
.1.3.6.1.4.1.140.625.105.1.15.32.54.98.49.101.57.56.54.98.98.50.57.10
0.54.55.48.100.56.98.101.101.97.55.48.53.57.99.49.51.56.98.97.99
STRING: MedRecServer_StartupEAR

Object ID:
.1.3.6.1.4.1.140.625.105.1.15.32.56.48.97.53.50.52.99.101.53.54.57.54
.52.52.99.54.48.55.54.100.102.49.54.97.98.52.48.53.98.100.100.49
STRING: MedRecServer_wl_management_internal2

...
```

The following example retrieves the name of all applications that have been deployed on all servers in the `medrec` domain.

```
java snmpwalk -c public@medrec localhost .1.3.6.1.4.1.140.625.105.1.15
```

The following example retrieves the name of all applications that have been deployed on a Managed Server named `MS1`.

```
java snmpwalk -c public@MS1 localhost .1.3.6.1.4.1.140.625.105.1.15
```

snmpgetnext

Returns a description of the managed object or object instance that immediately follows one or more OIDs that you specify. If you specify a tabular object, this command returns the first child managed object. If you specify a scalar object, this command returns the first instance of the object.

Instead of the recursive listing that the `snmpwalk` command provides, this command returns the description of only one managed object or instance whose OID is the next in sequence. You could string together a series of `snmpgetnext` commands to achieve the same result as the `snmpwalk` command.

Syntax

```
java snmpgetnext [-d] [-v (v1,v2)] [-c snmpCommunity] [-p snmpPort]
              [-t timeout] [-r retries] host OID [OID]...
```

Argument	Definition
OID [*OID*]...	One or more object IDs. Use a space to delimit multiple OIDs. You can specify OIDs for objects or instances. Start the values with '.'; otherwise, references are assumed to be relative to the standard MIB (`.1.3.6.1.2.1`), not the WebLogic Server MIB.

Example

The following example retrieves the name of an application that has been deployed on the Administration Server. The managed object for an application name is `applicationRuntimeName`, which is a scalar object and is a child of the `applicationRuntimeTable` object. (See WebLogic Server SNMP MIB Reference.)

```
java snmpgetnext localhost .1.3.6.1.4.1.140.625.105.1.15
```

If you invoke this command from a computer that is running the example MedRecServer, the command returns output similar to the following:

```
Response PDU received from /127.0.0.1, community: public
Object ID:
.1.3.6.1.4.1.140.625.105.1.15.32.49.102.98.97.100.97.102.99.57.48.50.
102.48.98.53.54.100.100.49.54.50.54.99.54.99.49.97.97.98.53.100.97
STRING: MedRecServer_uddiexplorer
```

To determine whether there are additional applications deployed on the Administration Server, you can use the output of the `snmpgetnext` command as input for an additional `snmpgetnext` command:

```
java snmpgetnext localhost
.1.3.6.1.4.1.140.625.105.1.15.32.49.102.98.97.100.97.102.99.57.48.50.102.
48.98.53.54.100.100.49.54.50.54.99.54.99.49.97.97.98.53.100.97
```

The command returns output similar to the following:

```
Response PDU received from /127.0.0.1, community: public
Object ID:
.1.3.6.1.4.1.140.625.105.1.15.32.54.98.49.101.57.56.54.98.98.50.57.10
0.54.55.48.100.56.98.101.101.97.55.48.53.57.99.49.51.56.98.97.99
STRING: MedRecServer_StartupEAR
```

The following example specifies two OIDs to retrieve the name of an application that has been deployed on the Administration Server **and** the name of a JDBC connection pool. The OIDs in the example command are for the `applicationRuntimeName` object, which is the name of an application, and `jdbcConnectionPoolRuntimeName`, which is the name of a JDBC connection pool.

```
java snmpgetnext localhost .1.3.6.1.4.1.140.625.105.1.15
.1.3.6.1.4.1.140.625.190.1.15
```

If you invoke this command from a computer that is running the example MedRecServer, the command returns output similar to the following:

```
Response PDU received from /127.0.0.1, community: public
Object ID:
.1.3.6.1.4.1.140.625.105.1.15.32.49.102.98.97.100.97.102.99.57.48.50.
102.48.98.53.54.100.100.49.54.50.54.99.54.99.49.97.97.98.53.100.97
STRING: MedRecServer_uddiexplorer
Object ID:
.1.3.6.1.4.1.140.625.190.1.15.32.53.53.49.48.50.55.52.57.57.49.99.102
.55.48.98.53.50.54.100.48.100.53.53.52.56.49.57.49.49.99.99.99
STRING: MedRecPool-PointBase
```

snmpget

Retrieves the value of one or more object instances. This command does not accept OIDs for managed objects.

Syntax

```
java snmpget [-d] [-v (v1,v2)] [-c snmpCommunity] [-p snmpPort]
             [-t timeout] [-r retries] host object-instance-OID
             [object-instance-OID]...
```

Argument	Definition
object-instance-OID [*object-instance-OID*]...	The object ID of an object **instance**. This command does not accept OIDs for managed objects. Start the value with '.'; otherwise, references are assumed to be relative to the standard MIB, not the WebLogic Server MIB.

Example

The following example retrieves the `serverRuntimeState` and `serverRuntimeListenPort` managed object instances for the Administration Server. Both of these objects are children of the `serverRuntimeTable` object. (See WebLogic Server SNMP MIB Reference.)

```
java snmpget localhost
.1.3.6.1.4.1.140.625.360.1.60.32.102.100.48.98.101.102.100.99.102.52.98.
97.48.48.49.102.57.53.51.50.100.102.53.55.97.101.52.56.99.99.97.99
.1.3.6.1.4.1.140.625.360.1.35.32.102.100.48.98.101.102.100.99.102.52.
98.97.48.48.49.102.57.53.51.50.100.102.53.55.97.101.52.56.99.99.97.99
```

If you invoke this command from a computer that is running the example MedRecServer, the command returns output similar to the following:

```
Response PDU received from /127.0.0.1, community: public
Object ID:
.1.3.6.1.4.1.140.625.360.1.60.32.102.100.48.98.101.102.100.99.102.52.
98.97.48.48.49.102.57.53.51.50.100.102.53.55.97.101.52.56.99.99.97.99
STRING: RUNNING
Object ID:
.1.3.6.1.4.1.140.625.360.1.35.32.102.100.48.98.101.102.100.99.102.52.
98.97.48.48.49.102.57.53.51.50.100.102.53.55.97.101.52.56.99.99.97.99
INTEGER: 7001
```

Commands for Testing Traps

Table 5-3 is an overview of commands that generate and receive traps for testing purposes.

Table 5-3 Overview of Commands for Retrieving Information about WebLogic Server

Command	Description
`snmptrapd`	Starts a daemon that receives traps and prints information about the trap. See "snmptrapd" on page 5-10.
`snmpv1trap`	Constructs an SNMPv1 trap and distributes it to the SNMP manager or trap daemon that is running on the specified host and listening on the specified port number. See "snmpv1trap" on page 5-10.

snmptrapd

Starts a daemon that receives traps and prints information about the trap.

Syntax

```
java snmptrapd [-d] [-c snmpCommunity] [-p TrapDestinationPort]
```

Argument	Definition
`-c snmpCommunity`	The community name that the SNMP agent (or `snmpv1trap` command) used to generate the trap. If you do not specify a value, the command assumes `-c public`.
`-p TrapDestinationPort`	The port number on which the trap daemon receives traps. If you do not specify a value, the command assumes `-p 162`.

Example

The following command starts a trap daemon and instructs it to listen for requests on port 165. The daemon runs in the shell until you kill the process or exit the shell:

```
java snmptrapd -p 165
```

If the command succeeds, the trap daemon returns a blank line with a cursor. The trap daemon waits in this state until it receives a trap, at which point it prints the trap.

snmpv1trap

Constructs an SNMPv1 trap and distributes it to the SNMP manager or trap daemon that is running on the specified host and listening on the specified port number.

As part of invoking this command, you specify the value for fields within the trap packet that you want to send. **The values that you specify must resolve to traps that are defined in the WebLogic Server MIB**. For information about WebLogic Server traps and the fields that trap packets require, refer to "Format of WebLogic Trap Notifications" in the *WebLogic SNMP Management Guide*.

Syntax

```
java snmpv1trap [-d] [-c snmpCommunity] [-p TrapDestinationPort]
          TrapDestinationHost .1.3.6.1.4.140.625
          agent-addr generic-trap specific-trap timestamp
          [OID {INTEGER | STRING | GAUGE | TIMETICKS | OPAQUE |
          IPADDRESS | COUNTER} value] ...
```

Argument	Definition
`-c snmpCommunity`	A community name for the trap. SNMP managers (or the trap daemon) can access the trap only if they are configured to use this community name. If you do not specify a value, the command assumes `-c public`.
`-p TrapDestinationPort`	The port number on which the SNMP manager or trap daemon is listening. If you do not specify a value, the command assumes `-p 162`.
TrapDestinationHost	The DNS name or IP address of the computer that hosts the SNMP manager or trap daemon.
`.1.3.6.1.4.140.625`	The value of the trap's `enterprise` field, which contains the beginning portion of the OID for all WebLogic Server traps.
agent-addr	The value of the trap's `agent address` field. This field is intended to indicate the computer on which the trap was generated. When using the `snmpv1trap` command to generate a trap, you can specify any valid DNS name or IP address.
generic-trap	The value of the trap's `generic trap type` field. For a list of valid values, refer to "Format of WebLogic Trap Notifications" in the *WebLogic SNMP Management Guide*.
specific-trap	The value of the trap's `specific trap type` field. For a list of valid values, refer to "Format of WebLogic Trap Notifications" in the *WebLogic SNMP Management Guide*.

Argument	Definition
timestamp	The value of the trap's `timestamp` field. This field is intended to indicate the length of time between the last re-initialization of the SNMP agent and the time at which the trap was issued. When using the `snmpv1trap` command to generate a trap, any number of seconds is sufficient.
`OID {INTEGER | STRING | GAUGE | TIMETICKS | OPAQUE | IPADDRESS | COUNTER}` *value*	(Optional) The value of the trap's `variable bindings` field, which consists of name–value pairs that further describe the trap notification. For each name–value pair, specify an OID, a value type, and a value. For example, a log message trap includes a `trapTime` binding to indicate the time at which the trap is generated. To include this variable binding in the test trap that you generate, specify the OID for the `trapTime` variable binding, the `STRING` keyword, and a string that represents the time: `.1.3.6.1.4.1.140.625.100.5 STRING "2:00 pm"`

Example

The following example generates a log message trap that contains the `trapTime` and `trapServerName` variable bindings. It broadcasts the trap through port 165. In the example:

- `6` is the generic trap value that specifies "other WebLogic Server traps."
- `60` is the specific trap value that WebLogic Server uses to identify log message traps.
- `.1.3.6.1.4.1.140.625.100.5` is the OID for the `trapTime` variable binding and `.1.3.6.1.4.1.140.625.100.10` is the OID for the `trapServerName` variable binding.

```
java snmpv1trap -p 165 localhost .1.3.6.1.4.140.625 localhost 6 60 1000
.1.3.6.1.4.1.140.625.100.5 STRING "2:00 pm" .1.3.6.1.4.1.140.625.100.10
STRING localhost
```

The SNMP manager (or trap daemon) that is listening at port number `165` receives the trap. If the trap daemon is listening on `165`, it returns the following:

```
Trap received from: /127.0.0.1, community: public
Enterprise: .1.3.6.1.4.140.625
Agent: /127.0.0.1
TRAP_TYPE: 6
SPECIFIC NUMBER: 60
Time: 1000
```

```
VARBINDS:
Object ID: .1.3.6.1.4.1.140.625.100.5
STRING: 2:00 pm
Object ID: .1.3.6.1.4.1.140.625.100.10
STRING: localhost
```

Example: Using snmpv1trap to Send Traps to the Trap Daemon

To use the `snmpv1trap` command to generate WebLogic Server traps and receive them through the trap daemon:

1. Open a command prompt (shell) and invoke the following script:

   `WL_HOME\server\bin\setWLSEnv.sh` (or `setWLSEnv.cmd` on Windows)
   where `WL_HOME` is the directory in which you installed WebLogic Server.

2. To start the trap daemon, enter the following command:

   ```
   java snmptrapd
   ```

3. Open another shell and invoke the following script:

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

4. To generate a trap, enter the following command:

   ```
   java snmpv1trap localhost .1.3.6.1.4.140.625 localhost 6 60 1000
   ```

The `snmpv1trap` command generates a `serverStart` trap and broadcasts it through port 162.

In the shell in which the trap daemon is running, the daemon prints the following:

```
Trap received from: /127.0.0.1, community: public
Enterprise: .1.3.6.1.4.140.625
Agent: /127.0.0.1
TRAP_TYPE: 6
SPECIFIC NUMBER: 60
Time: 1000
VARBINDS:
```

Example: Using the WebLogic SNMP Agent to Send Traps to the Trap Daemon

To use WebLogic SNMP agent to generate WebLogic Server traps and receive them through the trap daemon:

1. Start the Administration Server for a domain and enable the SNMP agent.

   See "Configure the SNMP Agent" in the *Administration Console Online Help*.

2. Create a trap destination to represent the trap daemon. Configure the trap destination to use port `165`. Keep all other default settings that the Administration Console presents.

   See "Create Trap Destinations" in the *Administration Console Online Help*.

3. Open a command prompt (shell) and invoke the following script:

   `WL_HOME\server\bin\setWLSEnv.sh` (or `setWLSEnv.cmd` on Windows)
   where `WL_HOME` is the directory in which you installed WebLogic Server.

4. To start the trap daemon, enter the following command:

   ```
   java snmptrapd -p 165
   ```

5. Restart the Administration Server.

When the Administration Server starts, the SNMP agent generates a `serverStart` trap and broadcasts it through port 165.

In the shell in which the trap daemon is running, the daemon prints the following:

```
Trap received from: /127.0.0.1, community: public
Enterprise: .1.3.6.1.4.140.625
Agent: /127.0.0.1
TRAP_TYPE: 6
SPECIFIC NUMBER: 65
Time: 1000
VARBINDS:
```

Index

A

Administration commands, overview 2-20, 2-39, 2-52, 5-4, 5-9
Administration Console
  specifying private key password for use with SSL 4-17
Administration Server
  starting from command line 4-1
appc 3-29

C

CANCEL_SHUTDOWN, WebLogic Server command 2-20
Command-line interface
  administration commands overview 2-20, 2-39, 2-52, 5-4, 5-9
  command syntax and arguments 2-2, 5-2
  Mbean management commands overview 2-63, 2-80
configuration files
  archiving 4-4
  back up copies 4-4
CONNECT, WebLogic Server command 2-21, 2-23
Connection Pool Administration commands, overview 2-52
CREATE, WebLogic Server command 2-63
CREATE_POOL, WebLogic Server command 2-53
Creating Mbeans, CREATE command 2-63

D

DDInit 3-11
DELETE, WebLogic Server command 2-65
Deleting Mbeans, DELETE command 2-65
DESTROY_POOL, WebLogic Server command 2-56
DISABLE_POOL, WebLogic Server command 2-57

E

ENABLE_POOL, WebLogic Server command 2-58
EXISTS_POOL, WebLogic Server command 2-61

G

GET, WebLogic Server command 2-67
Getting help for a WebLogic Server command 2-43
Getting Mbean information, GET command 2-67

H

HELP, WebLogic Server command 2-43
host name verification
  default configuration 2-7
  using 2-7
Host Name Verifier
  disabling at start-up 4-16
  specifying a custom 4-16

I

INVOKE, WebLogic Server command 2-69

J

Java heap memory
    specifying minimum and maximum 4-6
JHTML 3-18
jhtml2jsp 3-18
JNDI naming tree
    list node bindings 2-44

L

LICENSES, WebLogic Server command 2-44
LIST, WebLogic Server command 2-44
listen ports, setting 4-9
Listening ports, verify 2-29
LOCK, WebLogic Server command 2-28

M

Mbean management commands, overview 2-63, 2-80

P

PING, WebLogic Server command 2-29

R

RESET_POOL, WebLogic Server command 2-59, 2-60
Resetting connection pools, RESET_POOL command 2-59, 2-60

S

Secure Sockets Layer (SSL)
    host name verification 2-7
server name
    specifying at startup 4-26
SERVERLOG, WebLogic Server command 2-47
SET, WebLogic Server command 2-74
Setting attribute values, SET command 2-74
SHUTDOWN, WebLogic Server command 2-30
SSL
    specifying private key password at server startup 4-17
SSL session caching
    indicating 4-17
system home directory, WebLogic
    specifying at startup 4-7

T

THREAD_DUMP, WebLogic Server command 2-49
Threads, view running 2-49

U

UNLOCK, WebLogic Server command 2-39

V

Verify WebLogic Server listening ports 2-29
VERSION, WebLogic Server command 2-50
Viewing server log files, SERVERLOG command 2-47

W

WebLogic Server
    licenses, viewing 2-44
    specifying user name of at startup 4-19
WebLogic Server commands
    administration commands overview 2-20, 2-39, 2-52, 5-4, 5-9
    CANCEL_SHUTDOWN 2-20
    CONNECT 2-21, 2-23
    connection pool commands overview 2-52
    CREATE 2-63
    CREATE_POOL 2-53

DELETE 2-65
DESTROY_POOL 2-56
DISABLE_POOL 2-57
ENABLE_POOL 2-58
EXISTS_POOL 2-61
GET 2-67
HELP 2-43
INVOKE 2-69
LICENSES 2-44
LIST 2-44
LOCK 2-28
Mbean management commands overview
2-63, 2-80
PING 2-29
RESET_POOL 2-59, 2-60
SERVERLOG 2-47
SET 2-74
SHUTDOWN 2-30
syntax and arguments 2-2, 5-2
THREAD_DUMP 2-49
UNLOCK 2-39
VERSION 2-50
`weblogic.Server` startup command 4-9
`WebLogicObjectName`
defined 2-71