



BEA WebLogic Server®

WebLogic Server Command Reference

Version 9.0 BETA
Revised: December 15, 2004

Copyright

Copyright © 2004-2005 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, BEA WebLogic Server, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic JRockit, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server Process Edition, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

Contents

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

Required Environment for the weblogic.Admin Utility	1-2
Syntax for Invoking the weblogic.Admin Utility	1-3
SSL Arguments	1-3
Using SSL to Secure Administration Requests: Main Steps	1-5
Specifying Trust for weblogic.Admin	1-5
Specifying Host Name Verification for weblogic.Admin	1-6
Connection Arguments	1-7
User Credentials Arguments	1-10
Specifying User Credentials	1-12
Examples of Providing User Credentials	1-13
Protocol Support	1-13
Example Environment	1-14
Exit Codes Returned by weblogic.Admin	1-14
Command for Storing User Credentials	1-15
STOREUSERCONFIG	1-15
Syntax	1-15
Configuring the Default Path Name	1-17
Creating User-Configuration and Key Files	1-17
Using a Single Key File for Multiple User-Configuration Files	1-18
Examples	1-18
Commands for Managing the Server Life Cycle	1-19

CANCEL_SHUTDOWN	1-20
Syntax	1-21
Example	1-21
DISCOVERMANAGEDSERVER	1-21
Syntax	1-22
Example	1-23
FORCESHUTDOWN	1-23
Syntax	1-24
Example	1-27
LOCK	1-28
Syntax	1-28
Example	1-28
RESUME	1-29
Syntax	1-29
Example	1-29
SHUTDOWN	1-30
Syntax	1-30
Example	1-34
START	1-35
Syntax	1-36
Example	1-36
STARTINSTANDBY	1-37
Syntax	1-37
Example	1-38
UNLOCK	1-39
Syntax	1-39
Example	1-39

Commands for Retrieving Information about WebLogic Server and Server Instances . 1-39

CONNECT	1-40
Syntax	1-41
Example.....	1-41
GETSTATE	1-42
Syntax	1-42
Example.....	1-43
HELP	1-43
Syntax	1-43
Example.....	1-43
LICENSES	1-44
Syntax	1-44
Example.....	1-44
LIST	1-44
Syntax	1-45
Example.....	1-45
PING	1-46
Syntax	1-46
Example.....	1-46
SERVERLOG	1-47
Syntax	1-47
Example.....	1-48
THREAD_DUMP.....	1-49
Syntax	1-49
Example.....	1-50
VERSION	1-50
Syntax	1-50
Example.....	1-50
Commands for Managing JDBC Connection Pools	1-52

CREATE_POOL	1-53
Syntax	1-53
Example	1-55
DELETE_POOL	1-56
Syntax	1-56
Example	1-56
DESTROY_POOL	1-57
Syntax	1-57
Example	1-57
SUSPEND_POOL	1-58
Syntax	1-58
Example	1-59
DISABLE_POOL	1-59
Syntax	1-59
Example	1-60
RESUME_POOL	1-60
Syntax	1-60
Example	1-61
ENABLE_POOL	1-61
Syntax	1-61
Example	1-62
TEST_POOL	1-62
Syntax	1-62
Example	1-63
RESET_POOL	1-63
Syntax	1-63
Example	1-64
SHUTDOWN_POOL	1-64

Syntax	1-64
Example.	1-65
EXISTS_POOL.	1-65
Syntax	1-65
Example.	1-66
Commands for Managing WebLogic Server MBeans.	1-66
Specifying MBean Types	1-66
MBean Management Commands	1-67
CREATE	1-68
Syntax	1-68
Example.	1-69
DELETE	1-70
Syntax	1-70
Example.	1-71
GET	1-72
Syntax	1-73
Example.	1-74
INVOKE	1-74
Syntax	1-75
Example.	1-76
QUERY	1-76
Syntax	1-77
Example.	1-78
SET	1-79
Syntax	1-79
Example.	1-81
Running Commands in Batch Mode	1-82
BATCHUPDATE	1-82

Syntax	1-83
Example	1-84
Commands for Working with Clusters	1-84
CLUSTERSTATE	1-85
Syntax	1-85
Example	1-86
MIGRATE	1-86
Syntax	1-87
Examples	1-88
STARTCLUSTER	1-88
Syntax	1-89
Example	1-89
STOPCLUSTER	1-90
Syntax	1-90
Example	1-90
VALIDATECLUSTERCONFIG	1-91
Syntax	1-91
Example	1-92

2. Using the WebLogic Server Java Utilities

appc	2-3
AppletArchiver	2-3
Syntax	2-3
autotype (deprecated)	2-4
BuildXMLGen	2-4
CertGen	2-4
Syntax	2-4
Example	2-5

ClientDeployer	2-6
clientgen	2-6
Conversion (deprecated)	2-7
der2pem	2-7
Syntax	2-7
Example	2-8
dbping	2-8
Creating a DB2 Package with dbping	2-8
Syntax	2-8
Example	2-10
ddcreate	2-11
DDInit	2-11
Limitations	2-12
Example	2-12
Deployer	2-14
ejbc	2-14
EJBGen	2-14
encrypt	2-14
Syntax	2-15
Examples	2-15
getProperty	2-16
Syntax	2-16
Example	2-16
host2ior	2-17
Syntax	2-17
ImportPrivateKey	2-17
Syntax	2-17
Example	2-18

jhtml2jsp	2-19
Syntax	2-20
jspc	2-20
logToZip	2-20
Syntax	2-20
Examples.	2-21
MBean Commands	2-21
MulticastTest.	2-21
Syntax	2-22
Example	2-23
myip	2-24
Syntax	2-24
Example	2-24
pem2der	2-24
Syntax	2-24
Example	2-24
pointbase	2-25
rmic	2-25
Schema	2-25
Syntax	2-25
Example	2-26
servicegen (deprecated)	2-26
SearchAndBuild	2-26
Example	2-26
showLicenses	2-27
Syntax	2-27
Example	2-27
source2wsdd (deprecated).	2-27

system	2-27
Syntax	2-28
Example.	2-28
ValidateCertChain	2-28
verboseToZip.	2-29
Syntax	2-29
Example.	2-29
wlappc	2-30
wlcompile	2-30
wlconfig.	2-30
wldeploy	2-30
wlpackage	2-30
wlserver.	2-30
writeLicense	2-31
Syntax	2-31
Examples.	2-31
wsdl2Service	2-33
wsdlgen (deprecated).	2-33
wspackage (deprecated)	2-33

3. weblogic.Server Command-Line Reference

Required Environment and Syntax for weblogic.Server	3-2
Environment	3-2
Setting the Classpath	3-2
Syntax	3-3
Default Behavior	3-3
weblogic.Server Configuration Options	3-4
JVM Parameters	3-5

Location of License and Configuration Data	3-6
Examples	3-8
Server Communication	3-9
SSL	3-14
Setting Additional SSL Attributes	3-17
Security	3-17
Message Output and Logging	3-22
Setting Logging Attributes	3-22
Other Server Configuration Options	3-23
Clusters	3-25
Using the weblogic.Server Command Line to Start a Server Instance	3-25
Using the weblogic.Server Command Line to Create a Domain	3-26
Verifying Attribute Values That Are Set on the Command Line	3-27

4. WebLogic SNMP Agent Command-Line Reference

Required Environment and Syntax for the SNMP Command-Line Interface	4-2
Environment	4-2
Common Arguments	4-2
Commands for Retrieving the Value of WebLogic Server Attributes	4-4
snmpwalk	4-4
Syntax	4-5
Example	4-5
snmpgetnext	4-6
Syntax	4-6
Example	4-7
snmpget	4-8
Syntax	4-8
Example	4-9

Commands for Testing Traps	4-9
snmpv1trap	4-10
Syntax	4-10
Example.....	4-12
snmptrapd	4-13
Syntax	4-13
Example.....	4-13
Example: Sending Traps to the Trap Daemon	4-13

Index

BETA

BETA

weblogic.Admin Command-Line Reference (Deprecated)

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. 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 1-82](#).

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

- [“Required Environment for the weblogic.Admin Utility” on page 1-2](#)

- “Syntax for Invoking the weblogic.Admin Utility” on page 1-3
- “Command for Storing User Credentials” on page 1-15
- “Commands for Managing the Server Life Cycle” on page 1-19
- “Commands for Retrieving Information about WebLogic Server and Server Instances” on page 1-39
- “Commands for Managing JDBC Connection Pools” on page 1-52
- “Commands for Managing WebLogic Server MBeans” on page 1-66
- “Running Commands in Batch Mode” on page 1-82
- “Commands for Working with Clusters” on page 1-84

For more information, see:

- “Overview of Deployment Tools” in *Deploying WebLogic Server Applications*. Describes deployment tools to help you configure and deploy applications.
- “Automating WebLogic Server Administration Tasks” in *WebLogic Scripting Tool*. Describes using WLST commands to automate typical domain and server configuration tasks.

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](http://e-docs.bea.com/wls/docs90/install/index.html). See <http://e-docs.bea.com/wls/docs90/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 “[Setting the Classpath](#)” on page 3-2.

3. If you want the weblogic.Admin utility to use a listen port that is reserved for administration traffic, you must configure a domain-wide administration port as described in “[Enabling the Domain-Wide Administration Port](#)” in the *Administration Console Online Help*.

The domain-wide administration port is secured by SSL. For information about using secured ports with the weblogic.Admin utility, see “[SSL Arguments](#)” on page 1-3.

Note: If a server instance is deadlocked, it can respond to `weblogic.Admin` commands only if you have enabled the domain-wide administration port. If you have not already enabled the domain-wide administration port, your only option is to shut down the server instance by killing the Java process that is running the server. You will lose all session data. For information on enabling the domain-wide administration port, see ["Enabling 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 1-3](#)
- [“User Credentials Arguments” on page 1-10](#)
- [“Protocol Support” on page 1-13](#)

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 1-1](#) describes all SSL arguments for the `weblogic.Admin` utility.

Table 1-1 SSL Arguments

Argument	Definition
<code>-Dweblogic.security.TrustKeyStore=DemoTrust</code>	<p>Causes <code>weblogic.Admin</code> to trust the CA certificates in the demonstration trust keystore (<code>WL_HOME\server\lib\DemoTrust.jks</code>).</p> <p>This argument is required if the server instance to which you want to connect is using the demonstration identity and certificates.</p> <p>By default, <code>weblogic.Admin</code> trusts only the CA certificates in the Java Standard Trust keystore (<code>SDK_HOME\jre\lib\security\cacerts</code>).</p>
<code>-Dweblogic.security.JavaStandardTrustKeystorePassPhrase=password</code>	<p>Specifies the password that was used to secure the Java Standard Trust keystore.</p> <p>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.</p> <p>By default, the Java Standard Trust keystore is not protected by a password.</p>
<code>-Dweblogic.security.CustomTrustKeyStoreFileName=filename</code> <code>-Dweblogic.security.TrustKeystoreType=CustomTrust</code>	<p>Causes <code>weblogic.Admin</code> to trust the CA certificates in a custom keystore that is located at <i>filename</i>. You must use both arguments to trust custom keystores.</p>
<code>-Dweblogic.security.CustomTrustKeystorePassPhrase=password</code>	<p>Specifies the password that was used to secure the custom keystore.</p> <p>You must use this argument only if the custom keystore is protected by a password.</p>

Table 1-1 SSL Arguments

Argument	Definition
<code>-Dweblogic.security.SSL.hostnameVerifier=classname</code>	Specifies the name of a custom Host Name Verifier class. The class must implement the <code>weblogic.security.SSL.HostnameVerifier</code> interface.
<code>-Dweblogic.security.SSL.ignoreHostnameVerification=true</code>	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.

See ["Using the SSL Protocol to Connect to WebLogic Server from weblogic.Admin"](#) in *Managing WebLogic Security*.

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 1-13.

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

See ["Specifying Trust for weblogic.Admin"](#) on page 1-5.

- (Optional) A host name verifier.

See ["Specifying Host Name Verification for weblogic.Admin"](#) on page 1-6.

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 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 1-2](#) 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.

BETA

Table 1-2 Connection Arguments

Argument	Definition
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	<p>The listen address and listen port of the server instance that runs the command.</p> <p>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.</p> <p>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.</p> <p>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 <code>-adminurl</code> 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 <i>Developing Manageable Applications with JMX</i>.</p> <p>To use a listen port that is not secured by SSL, the format is <code>-url [protocol://]listen-address:port</code></p> <p>To use a port that is secured by SSL, the format is <code>-url secure-protocol://listen-address:port</code></p> <p>If you have set up a domain-wide administration port, you must specify the administration port number: <code>-url secure-protocol://listen-address:domain-wide-admin-port</code></p> <p>For information about valid values for <code>protocol</code> and <code>secure-protocol</code>, see "Protocol Support" on page 1-13.</p> <p>For more information about the listen address and listen ports, see "-Dweblogic.ListenAddress=host" on page 3-12 and "-Dweblogic.ListenPort= portnumber" on page 3-12.</p> <p>For more information about the domain-wide administration port, see "Enabling the Domain-Wide Administration Port" in the <i>Administration Console Online Help</i>.</p> <p>The default value for this argument is <code>t3://localhost:7001</code>.</p>

Table 1-2 Connection Arguments (Continued)

Argument	Definition
<code>-adminurl</code> <code>[protocol://]Admin-Server-listen-address:listen-port</code>	<p>Enables the Administration Server to retrieve Local Configuration MBeans or Runtime MBeans for any server instance in the domain.</p> <p>For information about types of MBeans, see "Understanding WebLogic Server MBeans" in <i>Developing Manageable Applications with JMX</i>.</p> <p>For all commands other than the MBean commands, <code>-adminurl admin-address</code> and <code>-url admin-address</code> are synonymous.</p> <p>The <code>-adminurl</code> value must specify the listen address and listen port of the Administration Server.</p> <p>To use a port that is not secured by SSL, the format is <code>-adminurl [protocol]Admin-Server-listen-address:port</code>.</p> <p>To use a port that is secured by SSL, the format is <code>-adminurl secure-protocol://Admin-Server-listen-address:port</code></p> <p>If you have set up a domain-wide administration port, you must specify the administration port number: <code>-adminurl secure-protocol://Admin-Server-listen-address:domain-wide-admin-port</code></p> <p>For information about valid values for <code>protocol</code> and <code>secure-protocol</code>, see "Protocol Support" on page 1-13.</p> <p>There is no default value for this argument.</p>

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 1-3](#) to provide the user credentials of a WebLogic Server user who has permission to invoke the command.

Table 1-3 User Credentials Arguments

Argument	Definition
<code>-username username</code>	<p>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.</p> <p>For information about permissions for system administration tasks, see "Security Roles" in the <i>Securing WebLogic Resources</i> guide.</p>
<code>-password password</code>	<p>The password that is associated with the username.</p> <p>If you do not specify the <code>-password</code> argument, <code>weblogic.Admin</code> prompts you for a password.</p> <p>If <code>WL_HOME\server\bin</code> is specified in the <code>PATH</code> environment variable, <code>weblogic.Admin</code> 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 1-2.</p>
<code>-userconfigfile config-file</code>	<p>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.</p> <p>If you do not specify <code>-userconfigfile config-file</code>, and if you do not specify <code>-username username</code>, <code>weblogic.Admin</code> searches for a user-configuration file at the default path name. (See "STOREUSERCONFIG" on page 1-15.)</p>
<code>-userkeyfile admin-key</code>	<p>Specifies the name and location of the key file that is associated with the user-configuration file you specify.</p> <p>When you create a user-configuration file, the <code>STOREUSERCONFIG</code> 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.</p> <p>If you do not specify <code>-userkeyfile admin-key</code>, <code>weblogic.Admin</code> searches for a key file at the default path name. (See "STOREUSERCONFIG" on page 1-15.)</p>

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 1-15](#).

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 1-17](#).

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 1-17](#).

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 ["Configuring the 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 ["Enabling and Configuring the IIOP Protocol"](#) in the *Administration Console Online Help*.

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

If you use either the `-url` or `-adminurl` argument to specify a port that is secured by SSL, you must specify either `t3s` or `https`. See [“Using SSL to Secure Administration Requests: Main Steps” on page 1-5](#).

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 7001.
- The Administration Server uses the name of its host machine, AdminHost, as its listen address. For more information about the listen address and listen ports, see [“-Dweblogic.ListenAddress=host” on page 3-12](#) and [“-Dweblogic.ListenPort=portnumber” on page 3-12](#).
- 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 1-12](#).

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
<code>-userconfigfile</code> <i>config-file</i>	<p>Specifies a file pathname at which the <code>STOREUSERCONFIG</code> command creates a user-configuration file. The pathname can be absolute or relative to the directory from which you enter the command.</p> <p>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.</p> <p>If you do not specify this option, <code>STOREUSERCONFIG</code> does the following:</p> <ul style="list-style-type: none"> ■ 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 1-17. ■ To determine the file name, it prepends your operating-system username to the string <code>-WebLogicConfig.properties</code>. For example, <code>username-WebLogicConfig.properties</code>. You can use Java options to specify a different username. See “Configuring the Default Path Name” on page 1-17.
<code>-userkeyfile</code> <i>keyfile</i>	<p>Specifies a file pathname at which the <code>STOREUSERCONFIG</code> command creates a key file. The pathname can be absolute or relative to the directory from which you enter the command.</p> <p>If a file already exists at the specified pathname, <code>STOREUSERCONFIG</code> uses the existing key file to encrypt the new user-configuration file.</p> <p>If you do not specify this option, <code>STOREUSERCONFIG</code> does the following:</p> <ul style="list-style-type: none"> ■ 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 1-17. ■ To determine the file name, it prepends your operating-system username to the string <code>-WebLogicKey.properties</code>. For example, <code>username-WebLogicKey.properties</code>. You can use Java options to specify a different username. See “Configuring the Default Path Name” on page 1-17.
<code>-username</code> <i>username</i> <code>[-password</code> <i>password]</i>	<p>Specifies the username and password to encrypt. The <code>STOREUSERCONFIG</code> command does not verify that the username and password are valid WebLogic Server user credentials.</p> <p>If you omit the <code>-password password</code> argument, <code>STOREUSERCONFIG</code> prompts you to enter a password.</p>

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 1-12](#).

Commands for Managing the Server Life Cycle

Table 1-4 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*.

Table 1-4 Overview of Commands for Managing the Server Life Cycle

Command	Description
CANCEL_SHUTDOWN	(Deprecated) Cancels the SHUTDOWN command for the WebLogic Server that is specified in the URL. See “CANCEL_SHUTDOWN” on page 1-20.
DISCOVERMANAGEDSERVER	(Deprecated) Causes the Administration Server to re-establish its administrative control over Managed servers. See “DISCOVERMANAGEDSERVER” on page 1-21.
FORCESHUTDOWN	Terminates a server instance without waiting for active sessions to complete. See “FORCESHUTDOWN” on page 1-23.
LOCK	(Deprecated) Locks a WebLogic Server against non-privileged logins. Any subsequent login attempt initiates a security exception which may contain an optional string message. See “LOCK” on page 1-28.
RESUME	Makes a server available to receive requests from external clients. See “RESUME” on page 1-29.
SHUTDOWN	Gracefully shuts down a WebLogic Server. See “SHUTDOWN” on page 1-30.
START	Uses a configured Node Manager to start a Managed Server in the RUNNING state. See “START” on page 1-35.
STARTINSTANDBY	(Deprecated) Uses a configured Node Manager to start a Managed Server and place it in the STANDBY state. See “STARTINSTANDBY” on page 1-37.
UNLOCK	(Deprecated) Unlocks the specified WebLogic Server after a LOCK operation. See “UNLOCK” on page 1-39.

CANCEL_SHUTDOWN

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

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

See [“SHUTDOWN” on page 1-30](#) and [“UNLOCK” on page 1-39](#).

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 1-14](#).

DISCOVERMANAGEDSERVER

(Deprecated) 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
<code>-url</code> <code>[<i>protocol://</i>]<i>listen-address:listen-port</i></code>	<p>You must specify the listen address and listen port of the Administration Server.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and "Protocol Support" on page 1-13.</p>
<code>-serverName</code>	<p>Specifies a Managed Server that is currently running.</p> <p>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.</p>

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

Example

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

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

For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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:7001 -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 "[Enabling 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]
```

BETA

Argument	Definition
<code>-Dweblogic.system.BootIdentityFile=filename</code> <code>[-Dweblogic.RootDirectory=path]</code>	<p>Cause the command to retrieve encrypted user credentials from a boot identity file. See "Boot Identity Files" in <i>Managing Server Startup and Shutdown</i>.</p> <p>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.</p> <p>If you do not use the <code>-username</code> argument or a user configuration file to specify credentials (see "User Credentials Arguments" on page 1-10), the command retrieves user credentials from a boot properties file as follows:</p> <ul style="list-style-type: none"> If you invoke the command from a server's root directory, and if the server's root directory contains a valid <code>boot.properties</code> 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 <code>boot.properties</code>, the command can use a boot identity file if you include the following argument: <code>-Dweblogic.system.BootIdentityFile=filename</code> where <i>filename</i> 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: <code>-Dweblogic.system.BootIdentityFile=filename</code> <code>-Dweblogic.RootDirectory=path</code> where <i>filename</i> is the fully qualified pathname of a valid boot identity file and <i>path</i> 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 <code>-username</code> and <code>-password</code> arguments to provide user credentials. If you specify both <code>-Dweblogic.system.BootIdentityFile=filename</code> and <code>-username</code> and <code>-password</code>, the command uses the credentials specified in the <code>-username</code> and <code>-password</code> arguments.

Argument	Definition (Continued)
<i>targetServer</i>	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 <code>-url</code> argument.

Example

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

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

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

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

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

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

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

For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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

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

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

Instead of using the LOCK command, start a server in the `STANDBY` state. In this state, a server instance responds only to administrative requests over the domain-wide administration port. 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
<code>-url</code> <code>[protocol://]<i>listen-ad</i> <i>dress:listen-port</i></code>	<p>Specify the listen address and listen port of the server instance that you want to lock.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and "Protocol Support" on page 1-13.</p>
<code>"<i>stringMessage</i>"</code>	<p>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.</p>

Example

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

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

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

RESUME

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

For more information about server states, 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
<code>-url</code> <code>secure-protocol://listen-en-address:listen-port</code>	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 <code>-url</code> entry in Table 1-2 and " Protocol Support " on page 1-13.
<code>targetServer</code>	The name of the server to resume. If you do not specify a value, the command resumes the server that you specified in the <code>-url</code> 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 1-14.

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 [“Shutting Down a Server”](#) 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 stuck 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 [“Enabling 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]
```

BETA

Argument	Definition
<code>-Dweblogic.system.BootIdentityFile=<i>filename</i></code> <code>[-Dweblogic.RootDirectory=<i>path</i>]</code>	<p>Cause the command to retrieve encrypted user credentials from a boot identity file. See "Boot Identity Files" in <i>Managing Server Startup and Shutdown</i>.</p> <p>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.</p> <p>If you do not use the <code>-username</code> argument or a user configuration file to specify credentials (see "User Credentials Arguments" on page 1-10), the command retrieves user credentials from a boot properties file as follows:</p> <ul style="list-style-type: none"> If you invoke the command from a server's root directory, and if the server's root directory contains a valid <code>boot.properties</code> 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 <code>boot.properties</code>, the command can use a boot identity file if you include the following argument: <code>-Dweblogic.system.BootIdentityFile=<i>filename</i></code> where <i>filename</i> 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: <code>-Dweblogic.system.BootIdentityFile=<i>filename</i></code> <code>-Dweblogic.RootDirectory=<i>path</i></code> where <i>filename</i> is the fully qualified pathname of a valid boot identity file and <i>path</i> 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 <code>-username</code> and <code>-password</code> arguments to provide user credentials. If you specify both <code>-Dweblogic.system.BootIdentityFile=<i>filename</i></code> and <code>-username</code> and <code>-password</code>, the command uses the credentials specified in the <code>-username</code> and <code>-password</code> arguments.

Argument	Definition
<code>-ignoreExistingSessions</code>	<p>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 <code>config.xml</code> file. For more information, see "Controlling Graceful Shutdowns" in <i>Managing Server Startup and Shutdown</i>.</p> <p>By default, a graceful shutdown operation waits for HTTP sessions to complete or timeout.</p>
<code>-timeout seconds</code>	<p>The number of seconds subsystems have to complete in-flight work and suspend themselves.</p> <p>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.</p> <p>If you do not specify a value, the SHUTDOWN command defers to the Graceful Shutdown Timeout setting that is specified on the Servers—Control—Start/Stop page of the Administration Console. By default, this setting is 0, which means that the server will wait indefinitely for graceful shutdown to complete.</p>
<code>targetServer</code>	<p>The name of the server to shut down.</p> <p>If you do not specify a value, the command shuts down the server that you specified in the <code>-url</code> argument.</p>
<code>seconds</code>	(Deprecated) Number of seconds allowed to elapse between the invoking of this command and the shutdown of the server.
<code>"stringMessage"</code>	(Deprecated) Message, in double quotes, to be supplied in the message that is sent if a user tries to log in while the WebLogic Server is being shut down.

Example

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

```
java weblogic.Admin -url AdminHost:7001 -username weblogic
-passwd 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 1-14](#).

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
-passwd 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 [“Configuring a Machine”](#) in the *Administration Console Online Help*.

The Startup Mode field in the Administration Console determines whether a Managed Server starts in the `RUNNING` state or `STANDBY` state. See [“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
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	<p>Must specify the listen address and listen port of the domain's Administration Server.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>
<code>targetServer</code>	The name of the Managed Server to start in a RUNNING state.

Example

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

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

When you issue the command, the following occurs:

1. The Administration Server determines which machine `MedRecManagedServer` is configured to run on. It instructs the Node Manager that is running on that machine to start `MedRecManagedServer` 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 Server—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 1-14](#).

STARTINSTANDBY

(Deprecated) Starts a Managed Server using Node Manager.

In previous releases, 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 the current release, 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 [“Starting a Managed Server in the STANDBY State”](#) in *Managing Server Startup and Shutdown*.

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 [“Configuring a Machine”](#) in the *Administration Console Online Help*.
- The domain must be configured to use a domain-wide administration port as described in [“Enabling the Domain-Wide Administration Port”](#) in the *Administration Console Online Help*.

For more information about server states, 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
<code>-url</code> <code>secure-protocol://listen-address:listen-port</code>	You must specify the Administration Server and domain-wide administration port as follows: <code>t3s://Admin-Server-listen-address:domain-wide-admin-port</code> or <code>https://Admin-Server-listen-address:domain-wide-admin-port</code> For more information, see the <code>-url</code> entry in Table 1-2 and “ Protocol Support ” on page 1-13 .
<code>targetServer</code>	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
-passwd 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 ~~Server—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 ~~Machine—Monitoring~~ page.

For more information about the environment in which this example runs, see “[Example Environment](#)” on [page 1-14](#).

UNLOCK

(Deprecated) Unlocks the specified WebLogic Server after a [LOCK](#) operation.

This command is deprecated because the `LOCK` command is deprecated. Instead of `LOCK` and `UNLOCK`, use `STARTINSTANDY` and `RESUME`. For more information, see [“RESUME” on page 1-29](#).

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 1-5](#) 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 1-5 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 1-40 .
GETSTATE	Returns the current state of the specified WebLogic Server instance. See “GETSTATE” on page 1-42 .

Table 1-5 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 1-43 .
LICENSES	Lists the licenses for all WebLogic Server instances that are installed on a specific server. See “LICENSES” on page 1-44 .
LIST	Lists the bindings of a node in a server’s JNDI naming tree. See “LIST” on page 1-44 .
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 1-46 . For a similar command that returns information about all servers in a cluster, see “CLUSTERSTATE” on page 1-85 .
SERVERLOG	Displays the server log file generated on a specific server instance. See “SERVERLOG” on page 1-47 .
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 1-49 .
VERSION	Displays the version of the WebLogic Server software that is running on the machine specified by the value of <i>URL</i> . See “VERSION” on page 1-50 .

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
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	<p>Specify the listen address and listen port of the server instance to which you want to connect.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>
<code>count</code>	<p>Number of connections the <code>weblogic.Admin</code> utility makes to the specified server instance.</p> <p>By default, this command makes only one connection.</p>

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 1-14.

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 stuck 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 "[Enabling 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
<i>targetServer</i>	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 <code>-url</code> 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 1-14](#).

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 listening 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 weblogic 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 1-14](#).

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
<i>JNDIcontextName</i>	<p>The JNDI context for lookup, for example, weblogic, weblogic.ejb, javax.</p> <p>By default, the command lists the bindings immediately below the InitialContext of the specified server instance.</p>

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:7001 -username weblogic
    -password weblogic LIST
```

For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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:7001 -username weblogic
    -password weblogic LIST mail
```

If the command succeeds, it returns the following:

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

PING

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

For information on returning a description of all servers in a cluster, see [“CLUSTERSTATE” on page 1-85](#).

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 ["Enabling 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
<i>roundTrips</i>	Number of pings.
<i>messageLength</i>	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 1-14](#).

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 *Administration Console Online Help*.

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 "[Local Log Files and Domain Log Files](#)" in the *Administration Console Online Help*.

Syntax

```
java [ SSL Arguments ] weblogic.Admin
    [-url URL]
    [ User Credentials Arguments ]
    SERVERLOG [starttime [endtime]]
```

Argument	Definition
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	<p>Specify the listen address and listen port of the server instance for which you want to retrieve the local log file.</p> <p>If you use the <code>-url</code> argument to specify the Administration Server, the command returns the local log file of the Administration Server.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>
<code>starttime</code>	<p>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 <code>yyyy/mm/dd</code>. Time is indicated using a 24-hour clock. The start date and time are entered inside quotation marks, in the following format: <code>"yyyy/mm/dd hh:mm"</code></p> <p>By default, SERVERLOG returns up to 500 messages in chronological order starting from the beginning of the current log file.</p>
<code>endtime</code>	<p>Specifies the end of a time range and causes SERVERLOG to return up to 500 messages with a time stamp that is after <code>starttime</code> and before <code>endtime</code>. The date format is <code>yyyy/mm/dd</code>. Time is indicated using a 24-hour clock. The end date and time are entered inside quotation marks, in the following format: <code>"yyyy/mm/dd hh:mm"</code></p> <p>By default, SERVERLOG returns up to 500 messages in chronological order starting with the <code>starttime</code> value and ending with the time at which you issued the SERVERLOG command.</p>

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 1-14](#).

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 stuck 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 "[Enabling the Domain-Wide Administration Port](#)" in the *Administration Console Online Help*.

Note: The `THREAD_DUMP` command is supported only on Sun JVM and 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 1-14](#).

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
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	<p>Specify the listen address and listen port of a WebLogic Server instance.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>

Example

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

Commands for Retrieving Information about WebLogic Server and Server Instances

```
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 1-14](#).

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

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

BETA

Commands for Managing JDBC Connection Pools

[Table 1-6](#) 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 *Programming WebLogic JDBC* at <http://e-docs.bea.com/wls/docs90/jdbc/index.html> and "JDBC Connection Pools" in the *Administration Console Online Help*.

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 1-6 Overview of Commands for Managing JDBC Connection Pools

Command	Description
CREATE_POOL	Creates a connection pool. See "CREATE_POOL" on page 1-53 .
DELETE_POOL	Deletes the MBean and the configuration for the connection pool. Replaces DESTROY_POOL. See "DELETE_POOL" on page 1-56 .
DESTROY_POOL	(Deprecated) Closes all connections in the connection pool and deletes the connection pool configuration. Use DELETE_POOL instead. See "DESTROY_POOL" on page 1-57 .
SUSPEND_POOL	Disables the pool, suspending all operations on pool connections until the pool is re-enabled. Replaces DISABLE_POOL. See "SUSPEND_POOL" on page 1-58 .
DISABLE_POOL	(Deprecated) Temporarily disables a connection pool, preventing any clients from obtaining a connection from the pool. Use SUSPEND_POOL instead. See "DISABLE_POOL" on page 1-59 .
RESUME_POOL	Re-enables a connection pool after it has been suspended. Replaces ENABLE_POOL. See "RESUME_POOL" on page 1-60 .

Table 1-6 Overview of Commands for Managing JDBC Connection Pools

Command	Description
ENABLE_POOL	<p>(Deprecated) 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.</p> <p>Use RESUME_POOL instead.</p> <p>See “ENABLE_POOL” on page 1-61.</p>
TEST_POOL	<p>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.</p> <p>See “TEST_POOL” on page 1-62</p>
RESET_POOL	<p>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.</p> <p>See “RESET_POOL” on page 1-63.</p>
SHUTDOWN_POOL	<p>Closes all connections in the connection pool and undeploys the connection pool on all targets.</p> <p>See “SHUTDOWN_POOL” on page 1-64</p>
EXISTS_POOL	<p>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.</p> <p>See “EXISTS_POOL” on page 1-65.</p>

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	(Deprecated) Identifies the different access lists within <code>fileRealm.properties</code> in the server configuration directory. Paired name must be <code>dynaPool</code> .
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.
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 <code>allowShrinking = True</code> , 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.

Argument	Definition
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.
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.

DELETE_POOL

Removes all targets for the connection pool and then deletes the connection pool from the configuration.

Syntax

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

Argument	Definition
-url <i>[protocol://]listen-address:listen-port</i>	Optional. 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 <code>t3://localhost:7001</code> . For more information, see the <code>-url</code> entry in Table 1-2 and “ Protocol Support ” on page 1-13 .
-poolName	Required. The name of the connection pool to delete.

Example

The following command deletes a connection pool named `demoPool` on a managed server instance of WebLogic Server is running on a machine named `AdminHost`:

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

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

The pool `demoPool` was successfully deleted from the configuration.

DESTROY_POOL

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

This command is deprecated. Use `DELETE_POOL` or `SHUTDOWN_POOL` instead.

Syntax

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

Argument	Definition
-url [<i>protocol://</i>] <i>listen-address:listen-port</i>	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 <code>t3://localhost:7001</code> . For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.
<i>poolName</i>	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
```

SUSPEND_POOL

Disables the pool, suspending all operations on pool connections until the pool is re-enabled. With the `false` argument, if connections are in use, applications using a connection will get an exception. WebLogic Server preserves all connections in the connection pool exactly as they were before the connection pool was suspended. With the `true` argument (force suspend), applications that are currently using a connection are forcibly disconnected, and connections that were in use when the connection pool was suspended are recreated (closed and reopened). Any transaction on the connections that are closed are rolled back. WebLogic Server preserves all other connections exactly as they were before the connection pool was suspended.

Syntax

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

Argument	Definition
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	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 <code>t3://localhost:7001</code> . For more information, see the <code>-url</code> entry in Table 1-2 and “ Protocol Support ” on page 1-13 .
<code>-poolName</code>	Required. The name of the connection pool to suspend.
<code>false</code>	Applications using a connection will get an exception. WebLogic Server preserves all connections in the connection pool exactly as they were before the connection pool was suspended.
<code>true</code>	Force suspend—applications that are currently using a connection are forcibly disconnected, and connections that were in use when the connection pool was suspended are recreated (closed and reopened).

Example

The following command suspends a connection pool named `demoPool` on a WebLogic Server instance where the administration server is running on a machine named `AdminHost`:

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

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

```
Connection pool demoPool is suspended successfully.
```

DISABLE_POOL

(Deprecated) 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.

This command is deprecated. Use `SUSPEND_POOL` instead.

Syntax

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

Argument	Definition
<code>-url</code> <i>[protocol://]listen-address:listen-port</i>	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 <code>t3://localhost:7001</code> . For more information, see the <code>-url</code> entry in Table 1-2 and “ Protocol Support ” on page 1-13 .
<code>poolName</code>	Name of the connection pool

Argument	Definition
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 demoPool false
```

RESUME_POOL

Re-enables a connection pool that you disabled with the `SUSPEND_POOL` command. `RESUME_POOL` marks the connection pool as enabled and allows applications to use connections from the connection pool. If you suspended the connection pool with the `false` argument (not force suspend), all connections are preserved exactly as they were before the connection pool was suspended. Applications that had reserved a connection before the connection pool was suspended can continue JDBC operations exactly where they left off. If you suspended the connection pool with the `true` argument (force suspend), connections that were not in use when the connection pool was suspended are preserved exactly as they were before the suspension. Connections that *were* in use were closed and reopened. Applications that had reserved a connection no longer have a valid JDBC context.

Syntax

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

Argument	Definition
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	Optional. Specify the listen address and listen port of a WebLogic Server instance on which the connection pool has been deployed and is suspended. If you specify a secure listen port, you must also specify a secure protocol. If you do not specify a value, the command assumes <code>t3://localhost:7001</code> . For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13 .
<code>-poolName</code>	Required. The name of the connection pool to resume.

Example

The following command resumes (re-enables) a connection pool named `demoPool`:

```
java weblogic.Admin -url ManagedHost:8001 -username weblogic
    -password weblogic RESUME_POOL -poolName demoPool
```

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

```
Connection pool demoPool is resumed successfully.
```

ENABLE_POOL

(Deprecated) 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.

This command is deprecated. Use `RESUME_POOL` instead.

Syntax

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

Argument	Definition
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	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 <code>t3://localhost:7001</code> . For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13 .
<code>poolName</code>	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 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
```

Argument	Definition
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	<p>Specify the listen address and listen port of a WebLogic Server instance on which the connection pool has been deployed.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>
<code>poolName</code>	Name of the connection pool as listed in the configuration file (<code>config.xml</code>).

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 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
```

Argument	Definition
<code>-url</code> <code>[protocol: /]listen- address:li sten-port</code>	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 <code>t3://localhost:7001</code> . For more information, see the <code>-url</code> entry in Table 1-2 and “ Protocol Support ” on page 1-13 .
<code>poolName</code>	Name of the connection pool as listed in the configuration file (<code>config.xml</code>).

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 demoPool
```

SHUTDOWN_POOL

Shuts down the connection pool by closing all database connections. With the `false` argument, if applications are currently using any connections from the pool, the operation will fail. With the `true` argument (force shutdown), all applications that are currently using a connection from the connection pool are forcibly disconnected.

Syntax

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

Argument	Definition
<code>-url</code> <i>[protocol://]listen-address:listen-port</i>	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 <code>t3://localhost:7001</code> . For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.
<code>-poolName</code>	Required. The name of the connection pool to shut down.
<code>false</code>	If applications are currently using any connections from the pool, the operation will fail.
<code>true</code>	Force shutdown—all applications that are currently using a connection from the connection pool are forcibly disconnected..

Example

The following command shuts down a connection pool named `demoPool` on a WebLogic Server instance running on a machine named `AdminHost` and listening at port 7001:

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

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

```
Connection pool demoPool is shutdown successfully.
```

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*

Argument	Definition
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	<p>Specify the listen address and listen port of the Administration Server.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>
<code>poolName</code>	Name of connection pool.

Example

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

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

Commands for Managing WebLogic Server MBeans

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

- “[Specifying MBean Types](#)” on [page 1-66](#)
- “[MBean Management Commands](#)” on [page 1-67](#)

“[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 [WebLogic Server Javadoc](#) for the `weblogic.management.configuration` or `weblogic.management.runtime` packages.

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

1. Find the MBean's interface class and remove the MBean suffix from the class name. For an MBean that is an instance of the `weblogic.management.runtime.JDBCConnectionPoolRuntimeMBean`, use `JDBCConnectionPoolRuntime`.
2. For a Local Configuration MBean, append `Config` to the name. For example, for a Local Configuration MBean that is an instance of the `weblogic.management.configuration.JDBCConnectionPoolMBean` interface class, use `JDBCConnectionPoolConfig`. For the corresponding Administration MBean instance, use `JDBCConnectionPool`.

MBean Management Commands

[Table 1-7](#) is an overview of the MBean management commands.

Table 1-7 MBean Management Command Overview

Command	Description
CREATE	Creates an Administration MBean instance. This command cannot be used for Runtime MBeans and we recommend that you do not use it to create Local Configuration MBeans. See “CREATE” on page 1-68 .
DELETE	Deletes an MBean instance. See “DELETE” on page 1-70 .
GET	Displays properties of MBeans. See “GET” on page 1-72 .
INVOKE	Invokes management operations that an MBean exposes for its underlying resource. See “INVOKE” on page 1-74 .

Table 1-7 MBean Management Command Overview (Continued)

Command	Description
QUERY	Searches for MBeans whose <code>WebLogicObjectName</code> matches a pattern that you specify. See “QUERY” on page 1-76 .
SET	Sets the specified property values for the named MBean instance. This command cannot be used for Runtime MBeans. See “SET” on page 1-79 .

CREATE

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

If the command is successful, it returns OK.

When you use this command to create an Administration MBean instance, you must use the `-url` argument to specify the Administration Server. WebLogic Server populates the Administration MBean with default values and saves the MBean’s configuration in the domain’s `config.xml` file. For some types of Administration MBeans, WebLogic Server does not create the corresponding Local Configuration MBean replica until you restart the server instance that hosts the underlying managed resource. For example, if you create a `JDBCConnectionPool` Administration MBean to manage a JDBC connection pool on a Managed Server named `ManagedMedRecServer`, you must restart `ManagedMedRecServer` so that it can create its local replica of the `JDBCConnectionPool` Administration MBean that you created. For more information on MBean replication and the life cycle of MBeans, see ["Understanding WebLogic Server MBeans"](#) in *Developing Manageable Applications 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 [<i>protocol://</i>] <i>listen-address:listen-port</i>	<p>Specify the listen address and listen port of the Administration Server. You can create Administration MBeans only on the Administration Server.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes t3://localhost:7001.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p> <p>Although the CREATE command also supports the <code>-adminurl</code> argument, we recommend that you do not use CREATE to create Local Configuration MBeans.</p>
-name <i>name</i>	The name you choose for the MBean that you are creating.
-type <i>mbeanType</i>	The type of MBean that you are creating. For more information, see “ Specifying MBean Types ” on page 1-66.
-mbean <i>objectName</i>	<p>Fully qualified object name of an MBean in the <code>WebLogicObjectName</code> format. For example: “<i>domain:Type=type,Name=name</i>”</p> <p>For more information, see the Javadoc for <code>WebLogicObjectName</code>.</p>

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 1-14.

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
<pre>{-url [protocol://]listen-ad dress:listen-port} or {-adminurl [protocol://]Admin-Ser ver-listen-address:lis ten-port}</pre>	<p>To delete Administration MBeans, use <code>-url</code> to specify the Administration Server's listen address and listen port.</p> <p>To delete Runtime MBeans or Local Configuration MBeans, use one of the following:</p> <ul style="list-style-type: none"> <code>-url</code> to specify the listen address and listen port of the server instance on which you want to delete MBeans. <code>-adminurl</code> to delete instances of a Runtime or Local Configuration MBean type from all server instances in the domain. <p>For more information, see the <code>-url</code> and <code>-adminurl</code> entries in Table 1-3 and “Protocol Support” on page 1-13.</p>
<code>-type mbeanType</code>	Deletes all MBeans of the specified type. For more information, see “Specifying MBean Types” on page 1-66 .
<code>-mbean objectName</code>	<p>Fully qualified object name of an MBean in the <code>WebLogicObjectName</code> format. For example:</p> <p><code>"domain:Type=type,Name=name"</code></p> <p>For more information, see the Javadoc for <code>WebLogicObjectName</code>.</p>

Example

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

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

For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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

```
Ok
```

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

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

```
MedRec:Location=MedRecManagedServer,Name=myPool,  
Type=JDBCConnectionPoolConfig
```

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

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

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

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

GET

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

The output of the command is as follows:

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

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  
attribute2: 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-ad dress:listen-port} or {-adminurl [protocol://]Admin-Ser ver-listen-address:lis ten-port}	<p>To retrieve Administration MBeans, use <code>-url</code> to specify the Administration Server's listen address and listen port.</p> <p>To retrieve Runtime MBeans or Local Configuration MBeans, use one of the following:</p> <ul style="list-style-type: none"> <code>-url</code> to specify the listen address and listen port of the server instance on which you want to retrieve MBeans. <code>-adminurl</code> to retrieve instances of a Runtime or Local Configuration MBean type from all server instances in the domain. <p>For more information, see the <code>-url</code> and <code>-adminurl</code> entries in Table 1-3 and “Protocol Support” on page 1-13.</p>
<code>-type mbeanType</code>	Returns information for all MBeans of the specified type. For more information, see “Specifying MBean Types” on page 1-66.
<code>-mbean objectName</code>	<p>Fully qualified object name of an MBean in the <code>WebLogicObjectName</code> format:</p> <p><code>"domain:Type=type,Location=location,Name=name"</code></p> <p>For more information, see the Javadoc for <code>WebLogicObjectName</code>.</p>
<code>-pretty</code>	Places property-value pairs on separate lines.
<code>-property property</code>	<p>The name of the MBean property (attribute) or properties to be listed.</p> <p>Note: If property is not specified using this argument, all properties are displayed.</p>

Example

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

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

For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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 `getAttribute` and `setAttribute` 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-ad dress:listen-port} or {-adminurl [protocol://]Admin-Ser ver-listen-address:lis ten-port}	<p>To invoke operations for Administration MBeans, use -url to specify the Administration Server's listen address and listen port.</p> <p>To invoke operations for Runtime MBeans, use one of the following:</p> <ul style="list-style-type: none"> -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. <p>We recommend that you do not invoke operations for Local Configuration MBeans. Instead, invoke the operation on the corresponding Administration MBean.</p>
-type mbeanType	Invokes the operation on all MBeans of a specific type. For more information, see “Specifying MBean Types” on page 1-66 .
-mbean objectName	<p>Fully qualified object name of an MBean, in the WebLogicObjectName format:</p> <p><code>"domain:Type=type,Location=location,Name=name"</code></p> <p>For more information see the Javadoc for WebLogicObjectName.</p>
-method methodname	Name of the method to be invoked.
argument	<p>Arguments to be passed to the method call.</p> <p>When the argument is a String array, the arguments must be passed in the following format:</p> <p><code>"String1;String2;. . . "</code></p>

Example

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

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

If the command succeeds, it returns the following:

```
{MBeanName="MedRec:Location=MedRecServer,Name=MedRecPool,ServerRuntime=Med
RecServer,Type=JDBCConnectionPoolRuntime"}
```

Ok

For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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 Manageable Applications 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
<code>{-url [protocol://]listen-address:listen-port}</code>	To search for Administration MBean object names, use <code>-url</code> to specify the Administration Server's listen address and listen port.
or <code>{-adminurl [protocol://]Admin-Server-listen-address:listen-port}</code>	<p>To search for the object names of Local Configuration or Runtime MBeans, use one of the following:</p> <ul style="list-style-type: none"><code>-url</code> to specify the listen address and listen port of the server instance on which you want to search.<code>-adminurl</code> to search on all server instances in the domain. <p>For more information, see the <code>-url</code> and <code>-adminurl</code> entries in Table 1-3 and “Protocol Support” on page 1-13.</p>

Argument	Definition
-pretty	Places property-value pairs on separate lines.
-pattern	<p>A partial <code>WebLogicObjectName</code> for which the <code>QUERY</code> command searches. The value must conform to the following pattern:</p> <p><i>domain-name:property-list</i></p> <p>For the <i>domain-name</i> portion of the pattern, you can use the <code>*</code> character, which matches any character sequence. Because the server instance that you specify with the <code>-url</code> or <code>-adminurl</code> argument can access only the MBeans that belong to its domain, the <code>*</code> character is sufficient. For example, if you use <code>-url</code> to specify a server in the MedRec domain, <code>QUERY</code> can only return MBeans that are in the MedRec domain. It cannot search for MBeans in a domain named mydomain.</p> <p>For the <i>property-list</i> portion of the pattern, specify one or more components (property-value pairs) of a <code>WebLogicObjectName</code>. For a list of all <code>WebLogicObjectName</code> property-value pairs, See "WebLogic Server MBean Object Names" in <i>Developing Manageable Applications with JMX</i>. (For example, all <code>WebLogicObjectName</code>s include <code>Name=value</code> and <code>Type=value</code> property-value pairs.)</p> <p>You can specify these property-value pairs in any order.</p> <p>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 <code>*</code> wildcard character in the place of one or more property-value pairs. For example, <code>Name=Med*</code> is not valid, but <code>Name=MedRecServer, *</code> is valid.</p> <p>If you provide at least one property-value pair in the <i>property-list</i>, you can locate the wildcard anywhere in the given pattern, provided that the <i>property-list</i> is still a comma-separated list.</p>

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 1-14](#).

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:7001 -username weblogic
    -password weblogic QUERY -pattern *:Name=MedRecPool,*
```

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

SET

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

If the command is successful, it returns OK.

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

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

Syntax

```
java [ SSL Arguments ] weblogic.Admin
    [-url URL]
    [ User Credentials Arguments ]
    SET {-type mbeanType|-mbean objectName}
    -property property1 property1_value
    [-property property2 property2_value] . . .
```

Argument	Definition
<code>-url</code> <code>[protocol://]listen-address:listen-port</code>	<p>Specifies the listen address and listen port of the Administration Server. Only the Administration Server can access Administration MBeans.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p> <p>Although the <code>SET</code> command supports the <code>-adminurl</code>, we recommend that you do not use it to set values of Local Configuration MBeans.</p>
<code>-type mbeanType</code>	<p>Sets the properties for all MBeans of a specific type. For more information, see “Specifying MBean Types” on page 1-66.</p>
<code>-mbean</code> <code>objectName</code>	<p>Fully qualified object name of an MBean in the <code>WebLogicObjectName</code> format. For example:</p> <p><code>"domain:Type=type,Name=name"</code></p> <p>For more information, see the Javadoc for <code>WebLogicObjectName</code>.</p>

Argument	Definition
<code>-property</code> <code>property</code>	The name of the property to be set.
<code>property _value</code>	<p>The value to be set.</p> <ul style="list-style-type: none"> 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 <code>WebLogicObjectName</code> format. For example: <code>"domain:Type=type,Name=name"</code> For more information, see the Javadoc for <code>WebLogicObjectName</code>. 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: <code>"domain:Name=name,Type=type;domain:Name=name,Type=type"</code> When the property value is a String array, separate each string by a semicolon and surround the entire property value list with quotes: <code>"String1;String2;. . . "</code> When the property value is a String or String array, you can set the value to null by using either of the following: <code>-property property-name "</code> <code>-property property-name</code> For example, both <code>-property ListenAddress "</code> and <code>-property ListenAddress</code> set the listen address to null. If the property value contains spaces, surround the value with quotes: <code>"-Da=1 -Db=3"</code> For example: <code>SET -type ServerStart -property Arguments "-Da=1 -Db=3"</code> When setting the properties for a JDBC Connection Pool, you must pass the arguments in the following format: <code>"user:username;password:password;server:servername"</code>

Example

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

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

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

For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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 1-82](#).

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
<pre>{-url [protocol://]listen-address:listen-port} or {-adminurl [protocol://]Admin-Server-listen-address:listen-port}</pre>	<p>If the batch file contains commands that access Administration MBeans, use <code>-url</code> to specify the Administration Server's listen address and listen port.</p> <p>If the batch file contains commands that access Local Configuration or Runtime MBeans, use one of the following:</p> <ul style="list-style-type: none"> <code>-url</code> to specify the listen address and listen port of the server instance on which you want to access MBeans. <code>-adminurl</code> to access all Local Configuration or Runtime MBeans in the domain. <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> and <code>-adminurl</code> entries in Table 1-2 and “Protocol Support” on page 1-13.</p>
<pre>-batchfile fileLocation</pre>	<p>The name of a text file that contains a list of <code>weblogic.Admin</code> commands. If you use a relative pathname, the root context is the directory from which you issue the <code>weblogic.Admin BATCHUPDATE</code> command.</p> <p>The file must contain one or more commands, formatted as follows:</p> <pre>COMMAND-NAME arguments</pre> <p>Place each command on a separate line.</p> <p>Within the batch file, the <code>BATCHUPDATE</code> command ignores any line that begins with a <code>#</code> character.</p> <p>Note: Quoted MBean names are allowed in the batch file.</p>

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

Example

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

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

The following command invokes the commands in `commands.txt`:

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

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

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

For information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

Commands for Working with Clusters

[Table 1-8](#) 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 1-8 MBean Management Command Overview

Command	Description
CLUSTERSTATE	Returns the number and state of servers in a cluster. See “CLUSTERSTATE” on page 1-85.
MIGRATE	Migrates a JMS service or a JTA service from one server instance to another within a cluster. See “MIGRATE” on page 1-86.
STARTCLUSTER	Starts all servers in a cluster See “STARTCLUSTER” on page 1-88.
STOPCLUSTER	Forces all servers in a cluster to shut down. See “STOPCLUSTER” on page 1-90.
VALIDATECLUSTERCONFIG	Parses the domain’s configuration file and reports any discrepancies in all cluster-related elements. See “VALIDATECLUSTERCONFIG” on page 1-91.

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
<code>{-url [protocol://]listen-ad dress:listen-port}</code>	<p>Specify the listen address and listen port of any server instance that is currently active and that belongs to the cluster.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes t3://localhost:7001.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>
<code>-clusterName clusterName</code>	The name of the cluster as specified in the domain’s configuration file (config.xml).

Example

The following example returns information about a cluster:

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

For more information about the environment in which this example runs, see “[Example Environment](#)” on page 1-14.

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 "[Migration for Pinned Services](#)" 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 1-2 and " Protocol Support " on page 1-13.
-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: <ul style="list-style-type: none"> 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, <i>which could cause corruption of the transaction log or of JMS messages</i> .
-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 [“Configuring a Machine”](#) in the *Administration Console Online Help*.

The Startup Mode field in the Administration Console determines whether a Managed Server starts in the `RUNNING` state or `STANDBY` state. See [“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
<code>{-url</code> <code>[protocol://]listen-address:listen-port}</code>	<p>Specify the listen address and listen port of the Administration Server.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>
<code>-clusterName</code> <code>clusterName</code>	<p>The name of the cluster as specified in the domain’s configuration file (<code>config.xml</code>).</p>

Example

The following example starts a cluster:

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

For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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
<code>{-url [protocol://]listen-address:listen-port}</code>	<p>Specify the listen address and listen port of the Administration Server.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>
<code>-clusterName clusterName</code>	<p>The name of the cluster as specified in the domain’s configuration file (<code>config.xml</code>).</p>

Example

The following example stops a cluster:

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


For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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
<code>{-url [protocol://]listen-address:listen-port}</code>	<p>Specify the listen address and listen port of any active server in the domain, regardless of whether it belongs to a cluster.</p> <p>If you specify a secure listen port, you must also specify a secure protocol.</p> <p>If you do not specify a value, the command assumes <code>t3://localhost:7001</code>.</p> <p>For more information, see the <code>-url</code> entry in Table 1-2 and “Protocol Support” on page 1-13.</p>
<code>-configPath <i>pathname</i></code>	<p>The path and file name of the domain's configuration file. A relative pathname is resolved to the directory in which you issue the <code>VALIDATECLUSTERCONFIG</code> command.</p>

Example

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

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

For more information about the environment in which this example runs, see [“Example Environment” on page 1-14](#).

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.

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 [“Setting the Classpath” on page 3-2](#).

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/>.

- [“appe” on page 2-3](#)
- [“AppletArchiver” on page 2-3](#)
- [“autotype \(deprecated\)” on page 2-4](#)
- [“BuildXMLGen” on page 2-4](#)
- [“CertGen” on page 2-4](#)
- [“ClientDeployer” on page 2-6](#)

- “clientgen” on page 2-6
- “Conversion (deprecated)” on page 2-7
- “der2pem” on page 2-7
- “dbping” on page 2-8
- “DDInit” on page 2-11
- “Deployer” on page 2-14
- “ejbc” on page 2-14
- “EJBGen” on page 2-14
- “encrypt” on page 2-14
- “getProperty” on page 2-16
- “host2ior” on page 2-17
- “ImportPrivateKey” on page 2-17
- “jspc” on page 2-20
- “logToZip” on page 2-20
- “MBean Commands” on page 2-21
- “MulticastTest” on page 2-21
- “myip” on page 2-24
- “pem2der” on page 2-24
- “rmic” on page 2-25
- “Schema” on page 2-25
- “showLicenses” on page 2-27
- “source2wsdd (deprecated)” on page 2-27
- “system” on page 2-27
- “ValidateCertChain” on page 2-28
- “verboseToZip” on page 2-29

- “wlappc” on page 2-30
- “wlcompile” on page 2-30
- “wlconfig” on page 2-30
- “wldeploy” on page 2-30
- “wlpkg” on page 2-30
- “wlserver” on page 2-30
- “writeLicense” on page 2-31
- “wsdl2Service” on page 2-33
- “wsdlgen (deprecated)” on page 2-33
- “wspkg (deprecated)” on page 2-33

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](#) 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`. 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 the Applications”](#) in *Developing Applications with WebLogic Server*.

CertGen

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

Syntax

```
$ java utils.CertGen -cacert ca_cert_filename -cakey ca_key_filename
-cakeypass ca_key_password -selfsigned -certfile certfile -keyfile
privatekeyfile -keyfilepass keyfilepassword -strength keystrength -cn
commonname -ou orgunit -l locality -s state -c countrycode -subjectkeyid
-subjectkeyidformat format
```

Argument	Definition
<code>ca_cert_filename</code>	The name of the issuer's trusted CA certificate.
<code>ca_key_filename</code>	The private key associated with the trusted CA certificate.
<code>ca_key_password</code>	The password for the trusted CA certificate's private key.
<code>selfsigned</code>	Generates a self-signed certificate that can be used as a trusted CA certificate. If this argument is specified, the <code>ca_cert_filename</code> , <code>ca_key_filename</code> , and <code>ca_key_password</code> arguments should not be specified.
<code>certfile</code>	The name of the generated certificate file.

Argument	Definition
<i>privatekeyfile</i>	The name of the generated private key file.
<i>keyfilepassword</i>	The password for the private key.
<i>keystrength</i>	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. If you have the domestic version of WebLogic Server, you can choose 512-, 768-, or 1024-bit keys. The 1024-bit key is the default.
<i>commonname</i>	The name to be associated with the generated certificate.
<i>orgunit</i>	The name of the organization associated with the generated certificate.
<i>locality</i>	The name of a city or town.
<i>state</i>	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.
<i>countrycode</i>	Two-letter ISO code for your country. The code for the United States is US.
<i>subjectkeyid</i>	Specifies whether or not the generated certificate is to have a Subject Key identifier.
<i>format</i>	UTF-8 string or BAS E64 encoded byte array. The default is UTF-8.

Example

To generate a certificate:

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

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

Encoding

.....
.....
.....

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

Encoding

.....
.....
.....

Created Certificate files - testcert.der and testcert.pem

.....

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 a client JAR file. 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 the [Upgrade Planning Guide](#).

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
<i>derFile</i>	The name of the file to convert. The filename must end with a <code>.der</code> extension, and must contain a valid certificate in <code>.der</code> format.
<i>headerFile</i>	<p>The header to place in the PEM file. The default header is “-----BEGIN CERTIFICATE-----”.</p> <p>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:</p> <ul style="list-style-type: none">“-----BEGIN RSA PRIVATE KEY-----” for an unencrypted private key.“-----BEGIN ENCRYPTED PRIVATE KEY-----” for an encrypted private key. <p>Note: There must be a new line at the end of the header line in the file.</p>
<i>footerFile</i>	<p>The header to place in the PEM file. The default header is “-----END CERTIFICATE-----”.</p> <p>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:</p> <ul style="list-style-type: none">“-----END RSA PRIVATE KEY-----” for an unencrypted private key.“-----END ENCRYPTED PRIVATE KEY-----” for an encrypted private key. <p>Note: There must be a new line at the end of the header line in the file.</p>

Example

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

dbping

The `dbping` command-line utility tests the connection between a DBMS and your client machine via a JDBC driver. You must complete the installation of the driver before attempting to use this utility. For more information on how to install a driver, see 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
<i>DBMS</i>	<p>Varies by DBMS and JDBC driver:</p> <p>DB2B—WebLogic Type 4 JDBC Driver for DB2</p> <p>JCONN2—Sybase JConnect 5.5 (JDBC 2.0) driver</p> <p>JCONNECT—Sybase JConnect driver</p> <p>INFORMIXB—WebLogic Type 4 JDBC Driver for Informix</p> <p>MSSQLSERVER4—WebLogic jDriver for Microsoft SQL Server</p> <p>MSSQLSERVERB—WebLogic Type 4 JDBC Driver for Microsoft SQL Server</p> <p>ORACLE—WebLogic jDriver for Oracle</p> <p>ORACLEB—WebLogic Type 4 JDBC Driver for Oracle</p> <p>ORACLE_THIN—Oracle Thin Driver</p> <p>POINTBASE—PointBase Universal Driver</p> <p>SYBASEB—WebLogic Type 4 JDBC Driver for Sybase</p>
<i>[-d dynamicSections]</i>	<p>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.</p> <p>If the <i>-d</i> option is specified, the driver automatically sets <code>CreateDefaultPackage=true</code> and <code>ReplacePackage=true</code> on the connection and creates a DB2 package with the number of dynamic sections specified.</p>
<i>user</i>	<p>Valid database username for login. Use the same values you use with <code>isql</code>, <code>sqlplus</code>, or other SQL command-line tools.</p> <p>For DB2 with the <i>-d</i> option, the user must have <code>CREATE PACKAGE</code> privileges on the database.</p>

Argument	Definition
<i>password</i>	Valid database password for the user. Use the same values you use with <code>isql</code> or <code>sqlplus</code> .
<i>DB</i>	<p>Name and location of the database. Use the following format, depending on which JDBC driver you use:</p> <p>DB2B—Host:Port/DBName</p> <p>JCONN2—Host:Port/DBName</p> <p>JCONNECT—Host:Port/DBName</p> <p>INFORMIXB—Host:Port/DBName/InformixServer</p> <p>MSSQLSERVER4—Host:Port/DBName or [DBName@]Host[:Port]</p> <p>MSSQLSERVERB—Host:Port/DBName</p> <p>ORACLE—DBName (as listed in <code>tnsnames.ora</code>)</p> <p>ORACLEB—Host:Port/DBName</p> <p>ORACLE_THIN—Host:Port/DBName</p> <p>POINTBASE—Host[:Port]/DBName</p> <p>SYBASEB—Host:Port/DBName</p> <p>Where:</p> <ul style="list-style-type: none"> • <i>Host</i> is the name of the machine hosting the DBMS, • <i>Port</i> is port on the database host where the DBMS is listening for connections, and • <i>DBName</i> is the name of a database on the DBMS. • <i>InformixServer</i> 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 2-12](#).

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.

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

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

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

EJBInit

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

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

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

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

WebInit

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

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

EarInit

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

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

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

Limitations

- Generated XML representations of relations among entity beans are only accurate for one-to-one relations. For entity beans that already have descriptors, Builder does not refresh relations that have a “many” side.
- Support for generating descriptors for EJB 1.1 beans is not guaranteed; focus is on EJB 2.0

Example

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

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

```
Found 4 classes that implement the EnterpriseBean interface
```

```
Discovered module type for D:\dev\smarticket5\smarticket\bin\ejb_st.jar
Found EJB components. Initializing descriptors

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

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

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

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

Adding Entity bean 'CustomerEJB'

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

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

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

LocaleInfoEJB is a Stateless Session bean

Adding Session bean 'LocaleInfoEJB'

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

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

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

MovieInfoEJB is a Stateless Session bean

Adding Session bean 'MovieInfoEJB'

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

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

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

```
TicketSalesEJB is a Stateful Session bean
Adding Session bean 'TicketSalesEJB'
Writing descriptors
Building module with newly created descriptors
Finished building module
```

Deployer

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](#).

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

ejbc

Deprecated. See [ejbc](#) 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 in 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
<code>weblogic.RootDirectory</code>	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).
<code>weblogic.management.allowPasswordEcho</code>	Optional. Allows echoing characters entered on the command line. <code>weblogic.security.Encrypt</code> expects that no-echo is available; if no-echo is not available, set this property to true.
<code>password</code>	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}12hsIIIn56KKKs3

getProperty

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

Syntax

```
$ java utils.getProperty
```

Example

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

host2ior

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

Syntax

```
$ java utils.host2ior hostname port
```

ImportPrivateKey

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

Syntax

```
$ java utils.ImportPrivateKey keystore storepass storetype keypass alias  
certfile keyfile keyfilepass
```

Argument	Definition
<i>keystore</i>	The name of the keystore. A new keystore is created if one does not exist.
<i>storepass</i>	The password to open the keystore.
<i>storetype</i>	The type of keystore being used. The following options are available: <ul style="list-style-type: none">• Demo Identity and Demo Trust—The demonstration Identity and Trust keystores located in the <code>WL_HOME\server\lib</code> directory and configured by default and the <code>cacerts</code> file in the <code>JAVA_HOME\jre\lib\security</code> directory.• Custom Identity and Java Standard Trust—An Identity keystore you create and the trusted CAs defined in the <code>cacerts</code> file in the <code>JAVA_HOME\jre\lib\security</code> directory.• Custom Identity and Custom Trust—Identity and Trust keystores you create.• Custom Identity and Command-Line Trust—An Identity keystore you create and command-line arguments that specify the location of the Trust keystore. Use this option in a production environment when the Administration port is enabled and Managed servers are started on the command-line instead of by the Node Manager.
<i>keypass</i>	The password used to retrieve the private key file from the keystore.
<i>alias</i>	The name that is used to look up certificates and keys in the keystore.

Argument	Definition
<i>certfile</i>	The name of the certificate associated with the private key.
<i>keyfile</i>	The name of the file holding the protected private key.
<i>keyfilepass</i>	The password used to unlock the private key file and to protect the private key in the keystore

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:

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

```
$ java utils.CertGen mykeypass testcert testkey
```

Creating Domestic Key Strength - 1024

Encoding

.....
.....
.....

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

Encoding

.....
.....
.....

Created Certificate files - testcert.der and testcert.pem

.....

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

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

.....
.....

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

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

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

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

jhtml2jsp

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

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 target directory isn't specified, output is written to current directory.

jspc

(Deprecated) JSP-specific compiler task. Use [“apcc” on page 2-3](#).

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
<i>logfile</i>	Required. Fully-qualified pathname of the log file.

Argument	Definition
<i>codebase</i>	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).
<i>zipfile</i>	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, QUERY, and SET) to administer MBeans. See [Commands for Managing WebLogic Server MBeans](#) in *weblogic.Admin Command-Line Reference*.

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/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
<code>-n name</code>	Required. A name that identifies the sender of the sequenced messages. Use a different name for each test process you start.
<code>-a address</code>	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.)
<code>-p portnumber</code>	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
<code>-t timeout</code>	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.
<code>-s send</code>	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
<i>pemFile</i>	The name of the file to be converted. The filename must end with a <code>.pem</code> extension, and it must contain a valid certificate in <code>.pem</code> 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.

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 [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
<i>driverURL</i>	Required. URL for the JDBC driver.
<i>driverClass</i>	Required. Pathname of the JDBC driver class.
<i>-u username</i>	Optional. Valid username.
<i>-p password</i>	Optional. Valid password for the user.
<i>-verbose</i>	Optional. Prints SQL statements and database messages.
<i>SQLfile</i>	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. 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—itsself—forever. See [FileSet](http://ant.apache.org/manual/CoreTypes/fileset.html) 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
<i>license_location</i>	The fully qualified name of the directory where the <code>license.bea</code> 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. 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 * * * * *
1.1.6

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

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

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

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

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

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=US
Cert[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=corbal, 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=US
Certificate 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
<i>inputFile</i>	Required. Temporary file that contains the output of the application running in verbose mode.
<i>zipFileToCreate</i>	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 “[appc](http://e-docs.bea.com/wls/docs90/ejb/appc_ejbc.html#appc)” at http://e-docs.bea.com/wls/docs90/ejb/appc_ejbc.html#appc.

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 “[Compiling Applications Using wlcompile](#)” 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 Applications Using wldeploy](#)” 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 “[Packaging Applications Using wlpackage](#)” 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	Required. Sends the output to <code>stdout</code> instead of <code>writeLicense.txt</code> .
-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 NT Output

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

Using the WebLogic Server Java Utilities

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

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

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

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

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

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

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

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

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

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 [wsdl2Service](#) 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. See [“Ant Task Reference”](#) in *Programming Web Services for Weblogic Server*.

wspackge (deprecated)

Use the Web Services `wspackge` 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. See [“Ant Task Reference”](#) in *Programming Web Services for Weblogic Server*.

BETA

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 3-2
- “Default Behavior” on page 3-3
- “`weblogic.Server` Configuration Options” on page 3-4
- “Using the `weblogic.Server` Command Line to Start a Server Instance” on page 3-25
- “Using the `weblogic.Server` Command Line to Create a Domain” on page 3-26
- “Verifying Attribute Values That Are Set on the Command Line” on page 3-27

For information about using scripts to start an instance of WebLogic Server, see “[Starting Administration Servers](#)” and “[Starting Managed Servers From a WebLogic Server 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 *Designing and Configuring WebLogic Server Environments*.

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. Add WebLogic Server classes to the `CLASSPATH` environment variable, as described in “Setting the Classpath” on page 3-2.
3. Include a Java Virtual Machine (JVM) in your `PATH` environment variable. You can use any JVM that is listed in the Supported Configurations page at <http://e-docs.bea.com/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.

Setting the Classpath

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

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

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

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

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

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

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

Depending on which WebLogic Server release, service pack, or patch that you have installed, this file might not exist on your system. Regardless of whether the file currently exists on your system, we recommend that you include

`WL_HOME/server/lib/weblogic_sp.jar` in your classpath to ensure compatibility with any updates. You must add this file to the classpath before you add `weblogic.jar`.

- `WL_HOME/server/lib/weblogic.jar`
- If you use the trial version of PointBase, an all-Java database management system, then include the following files:

`WL_HOME/common/eval/pointbase/lib/
pbserver44.jar` and `pbclient44.jar`

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

`WL_HOME/server/lib/wlepool.jar`
`WL_HOME/server/lib/wleorb.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 3-2](#), when you enter the command `java weblogic.Server` with no options, WebLogic Server does the following:

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

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

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

If it does not find a server named `myserver`, WebLogic Server exits the `weblogic.Server` process and generates an error message.

3. If there is no `config.xml` file in the current directory, WebLogic Server asks if you want to create a domain and server instance. If you answer yes, WebLogic Server does the following:

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

WebLogic Server also creates a `config.xml.booted` file, which is a copy of the `config.xml` file in its state just after the server successfully boots. If the `config.xml` becomes corrupted, you can boot the server with this `config.xml.booted` file.

- 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 files named `DefaultAuthenticatorInit.ldif` and `SerializedSystemIni.dat`.

WebLogic Server also encrypts and stores your username and password in a `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, `startmydomain.cmd` and `startmydomain.sh`, that you can use to start subsequent instantiations of the server. You can use a text editor to modify startup options such as whether the server starts in production mode or development mode. The `startmydomain` script contains comments that describe each option.

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 following attributes of a server instance:

- "[JVM Parameters](#)" on page 3-5
- "[Location of License and Configuration Data](#)" on page 3-6
- "[Server Communication](#)" on page 3-9
- "[SSL](#)" on page 3-14

- “Security” on page 3-17
- “Message Output and Logging” on page 3-22
- “Other Server Configuration Options” on page 3-23
- “Clusters” on page 3-25

Unless you are creating a new domain as described in [“Using the weblogic.Server Command Line to Create a Domain”](#) on page 3-26, 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, the `weblogic.Admin` command, or WebLogic Scripting Tool (WLST) to modify the `config.xml` file.

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. For more information, see [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 3-27.

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 3-1 Frequently Used Options for Setting JVM Parameters

Option	Description
<code>-Xms</code> and <code>-Xmx</code>	<p>Specify the minimum and maximum values (in megabytes) for Java heap memory.</p> <p>For example, you might want to start the server with a default allocation of 200 megabytes of Java heap memory to the WebLogic Server. To do so, you can start the server with the <code>java -Xms200m</code> and <code>-Xmx200m</code> options.</p> <p>For best performance it is recommended that the minimum and maximum values be the same so that the JVM does not resize the heap.</p> <p>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.</p>
<code>-classpath</code>	<p>The minimum content for this option is described under “Setting the Classpath” on page 3-2.</p> <p>Instead of using this argument, you can use an environment variable named <code>CLASSPATH</code> to specify the classpath.</p>
<code>-client</code> <code>-server</code>	<p>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.</p>

Location of License and Configuration Data

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

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

Option	Description
<code>-Dbea.home=bea_home</code>	<p>Specifies the location of the BEA home directory, which contains licensing and other essential information.</p> <p>By default, <code>weblogic.Server</code> determines the location of the BEA home directory based on values in the classpath.</p>
<code>-Dweblogic.RootDirectory=path</code>	<p>Specifies the server's root directory.</p> <p>By default, the root directory is the directory from which you issue the start command. For more information, see "A Server's Root Directory" in the <i>Administration Console Online Help</i>.</p>
<code>-Dweblogic.ConfigFile=file_name</code>	<p>Specifies a configuration file for your domain. The <code>file_name</code> value must see a valid XML file that conforms to the schema as defined in the "BEA WebLogic Server Configuration Reference."</p> <p>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 <code>-Dweblogic.RootDirectory</code>.</p> <p>The <code>file_name</code> value cannot contain a pathname component. For example, the following value is invalid:</p> <pre>-Dweblogic.ConfigFile=c:\mydir\myfile.xml</pre> <p>Instead, use the following arguments:</p> <pre>-Dweblogic.RootDirectory=c:\mydir -Dweblogic.ConfigFile=myfile.xml</pre> <p>If you do not specify this value, the default is <code>config.xml</code> in the server's root directory.</p>

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

Option	Description
<code>-Dweblogic.management.Generat eDefaultConfig=true</code>	<p>Prevents the <code>weblogic.Server</code> class from prompting for confirmation when creating a <code>config.xml</code> file.</p> <p>Valid only if you invoke <code>weblogic.Server</code> in an empty directory. See “Default Behavior” on page 3-3.</p>
<code>-Dweblogic.Domain=<i>domain</i></code>	<p>Specifies the name of the domain.</p> <p>If you are using <code>weblogic.Server</code> to create a domain, you can use this option to give the domain a specific name.</p> <p>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:</p> <p><code>.../config/domain/config.xml</code></p> <p>where <i>domain</i> is the name of the domain.</p> <p>If your domain’s configuration file conforms to that pathname, and if you invoke the <code>weblogic.Server</code> command from a directory other than <code>config/domain</code>, you can include the</p> <p><code>-Dweblogic.Domain=<i>domain</i></code> argument to cause WebLogic Server to search for a <code>config.xml</code> file in a pathname that matches <code>con fig/domain/config.xml</code>.</p>

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

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 3-27](#).

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
```

Server Communication

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

BETA

Table 3-3 Options for Configuring Server Communication

Option	Description
<code>-Dweblogic.management.server=[protocol://]Admin-host:port</code>	<p data-bbox="606 388 1231 470">Starts a server instance as a Managed Server and specifies the Administration Server that will configure and manage the server instance.</p> <p data-bbox="606 489 1251 807">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 <code>-Dweblogic.management.server</code> 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 <code>-Dweblogic.management.server</code> option in the start command.</p> <p data-bbox="606 824 1251 933">For <i>protocol</i>, 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.</p> <p data-bbox="606 958 1251 1067">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.</p> <p data-bbox="606 1086 1251 1142">For <i>Admin-host</i>, specify localhost or the DNS name or IP address of the machine where the Administration Server is running.</p> <p data-bbox="606 1159 1251 1241">For <i>port</i>, specify the Administration Server's listen port. If you set up the domain-side administration port, <i>port</i> must specify the domain-wide administration port.</p> <p data-bbox="606 1258 1251 1340">For more information on configuring a connection to the Administration Server, see "Configuring a Connection to the Administration Server" in <i>Managing Server Startup and Shutdown</i>.</p>

Table 3-3 Options for Configuring Server Communication

Option	Description
<code>-Dweblogic.ListenAddress=host</code>	<p>Specifies the address at which this server instance listens for requests. The <i>host</i> value must be either the DNS name or the IP address of the computer that is hosting the server instance.</p> <p>This startup option overrides any listen address value specified in the <code>config.xml</code> file. The override applies to the current server instantiation; it does not modify the value in the <code>config.xml</code> file. Use the Administration Console, the <code>weblogic.Admin</code> command, or WLST to modify the <code>config.xml</code> file.</p> <p>For more information, see "Configuring the Listen Address" in the <i>Administration Console Online Help</i> and "Editing Configuration MBeans" in <i>WebLogic Scripting Tool</i>.</p>
<code>-Dweblogic.ListenPort=portnumber</code>	<p>Enables and specifies the plain-text (non-SSL) listen port for the server instance.</p> <p>This startup option overrides any listen port value specified in the <code>config.xml</code> file. The override applies to the current server instantiation; it does not modify the value in the <code>config.xml</code> file. Use the Administration Console, the <code>weblogic.Admin</code> command, or WLST to modify the <code>config.xml</code> file.</p> <p>The default listen port is 7001.</p> <p>For more information, see "Configuring the Listen Ports" in the <i>Administration Console Online Help</i> and "Editing Configuration MBeans" in <i>WebLogic Scripting Tool</i>.</p>

Table 3-3 Options for Configuring Server Communication

Option	Description
<code>-Dweblogic.ssl.ListenPort=portnumber</code>	<p>Enables and specifies the port at which this WebLogic Server instance listens for SSL connection requests.</p> <p>This startup option overrides any SSL listen port value specified in the <code>config.xml</code> file. The override applies to the current server instantiation; it does not modify the value in the <code>config.xml</code> file. Use the Administration Console, the <code>weblogic.Admin</code> command, or WLST to modify the <code>config.xml</code> file.</p> <p>The default SSL listen port is 7002.</p> <p>For more information, see "Configuring the Listen Ports" in the <i>Administration Console Online Help</i> and "Editing Configuration MBeans" in <i>WebLogic Scripting Tool</i>.</p>
<code>-Dweblogic.management.discover={true false}</code>	<p>Determines whether an Administration Server recovers control of a domain after the server fails and is restarted.</p> <p>A <code>true</code> value causes an Administration Server to communicate with all known Managed Servers and inform them that the Administration Server is running.</p> <p>A <code>false</code> value prevents an Administration Server from communicating with any Managed Servers that are currently active in the domain.</p> <p>Caution: Specify <code>false</code> for this option only in the development environment of a single server. Specifying <code>false</code> can cause server instances in the domain to have an inconsistent set of deployed modules.</p> <p>For information on re-establishing administrative control over Managed Servers after an Administration Server has already started, see "DISCOVERMANAGEDSERVER" on page 1-21.</p>

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 3-27.

SSL

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

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

BETA

Table 3-4 Options for Configuring SSL

Option	Description
<code>-Dweblogic.security.SSL.ignoreHostnameVerification=true</code>	<p>Disables host-name verification, which enables you to use the demonstration digital certificates that are shipped with WebLogic Server.</p> <p>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.</p> <p>If you disable host name verification, either by using this option or by modifying the server's configuration in the <code>config.xml</code> file, the server instance does not verify host names when it is in the role of SSL client.</p> <p>Note: BEA does not recommend using the demonstration digital certificates or turning off host name verification in a production environment.</p> <p>This startup option overrides any Host Name Verification setting in the <code>config.xml</code> file. The override applies to the current server instantiation; it does not modify the value in the <code>config.xml</code> file. Use the Administration Console or the <code>weblogic.Admin</code> command to modify the <code>config.xml</code> file.</p> <p>For more information, see "Using Hostname Verification" in the <i>Managing WebLogic Security</i> guide.</p>
<code>-Dweblogic.security.SSL.HostnameVerifier=hostnameverifierimplmentation</code>	<p>Specifies the name of a custom Host Name Verifier class. The class must implement the <code>weblogic.security.SSL.HostnameVerifier</code> interface.</p>

Table 3-4 Options for Configuring SSL

Option	Description
<code>-Dweblogic.security.SSL. sessionCache.size= sessionCacheSize</code> <code>-Dweblogic.security.SSL. sessionCache.ttl= sessionCacheTimeToLive</code>	<p>Modifies the default server-session caching size and time-to-live for SSL session caching.</p> <p>The <i>sessionCacheSize</i> value specifies the number of items in session cache and the <i>sessionCacheTimeToLive</i> value specifies (in seconds) the session cache time-to-live.</p> <p>For <i>sessionCache.size</i>:</p> <ul style="list-style-type: none"> • The minimum value is 1 • The maximum value is 65537 • The default value is 211 <p>For <i>sessionCache.ttl</i>:</p> <ul style="list-style-type: none"> • The minimum value is 1 • The maximum value is <code>Integer.MAX_VALUE</code> • The default value is 600
<code>-Dweblogic.management. pkpassword=<i>pkpassword</i></code>	<p>Specifies the password for retrieving SSL private keys from an encrypted flat file.</p> <p>Use this option if you store private keys in an encrypted flat file.</p>
<code>-Dweblogic.security.SSL. trustedCAKeyStore=<i>path</i></code>	<p>Deprecated and ignored by default.</p> <p>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 <i>path</i> value must be a relative or qualified name to the Sun JKS keystore file (contains a repository of keys and certificates).</p> <p>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 <code>JAVA_HOME\jre\lib\security\cacerts</code>.</p> <p>We recommend that you do not use the demonstration certificate authorities in any type of production deployment.</p> <p>For more information, see "Configuring SSL" in the <i>Managing WebLogic Security</i> guide.</p>

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 3-27](#).

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 3-27](#).

Security

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

Table 3-5 Options for General Security Parameters

Option	Description
<code>-Dweblogic.management. username=username</code>	<p>Specifies the username under which the server instance will run.</p> <p>The username must belong to a role that has permission to start a server. For information on roles and permissions, see "Security Roles" in the <i>Securing WebLogic Resources</i> guide.</p> <p>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 <i>Managing Server Startup and Shutdown</i>.</p>
<code>-Dweblogic.management. password=password</code>	<p>Specifies the user password.</p> <p>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 <i>Managing Server Startup and Shutdown</i>.</p>
<code>-Dweblogic.system. StoreBootIdentity=true</code>	<p>Creates a <code>boot.properties</code> 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.</p> <p>Do not specify this argument in a server's <code>ServerStartMBean</code> (Remote Startup tab in the Administration Console). For more information, see "Specifying User Credentials When Starting a Server with the Node Manager" in <i>Managing Server Startup and Shutdown</i>.</p> <p>BEA recommends that you do not add this argument to a startup script. Instead, use it only when you want to create a <code>boot.properties</code> file.</p> <p>For more information, see "Boot Identity Files" in <i>Managing Server Startup and Shutdown</i>.</p>

Table 3-5 Options for General Security Parameters

Option	Description
<code>-Dweblogic.system. BootIdentityFile=filename</code>	<p>Specifies a boot identity file that contains a username and password. The <i>filename</i> value must be the fully qualified pathname of a valid boot identity file. For example:</p> <pre>-Dweblogic.system.BootIdentityFile=C:\BEA\ wlserver8.1\user_config\mydomain\myidentity.pr op</pre> <p>If you do not specify a filename, a server instance or the <code>weblogic.Admin SHUTDOWN</code> and <code>FORCESHUTDOWN</code> commands use the <code>boot.properties</code> file in the server's root directory.</p> <p>If there is no boot identity file:</p> <ul style="list-style-type: none"> • When starting a server, the server instance prompts you to enter a username and password. • When using the <code>weblogic.Admin SHUTDOWN</code> and <code>FORCESHUTDOWN</code> commands, you must use the <code>-username</code> and <code>-password</code> arguments to provide user credentials.
<code>-Dweblogic.system. RemoveBootIdentity=true</code>	<p>Removes the boot identity file after a server starts.</p>
<code>-Dweblogic.security.anonymous UserName=name</code>	<p>Assigns a user ID to anonymous users. By default, all anonymous users are identified with the string <code><anonymous></code>.</p> <p>To emulate the security behavior of WebLogic Server 6.x, specify <code>guest</code> for the <i>name</i> value and create a user named <code>guest</code> in your security realm.</p> <p>For more information, see “Users and Groups” in the <i>Securing WebLogic Resources</i> guide.</p>

Table 3-5 Options for General Security Parameters

Option	Description
<code>-Djava.security.manager</code> <code>-Djava.security.policy[=]<i>filename</i></code>	<p>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.</p> <p>To use the WebLogic Server sample policy file, specify <code>WL_HOME\server\lib\weblogic.policy</code>.</p> <p>Using <code>-Djava.security.policy=<i>filename</i></code> causes the policy file to override any default security policy. A single equal sign (=) causes the policy file to be appended to an existing security policy.</p> <p>For more information, see “Using the Java Security Manager to Protect WebLogic Resources” in the <i>Programming WebLogic Security</i> guide.</p>
<code>-Dweblogic.security.fullyDelegateAuthorization=true</code>	<p>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.</p> <p>Use this option when starting WebLogic Server to override this problem.</p> <p>This startup option does not work with EJBs or EJB methods that use <code><unchecked></code> or <code><restricted></code> tags or Web applications that do not have a role-name specified in the <code><auth-constraint></code> tag.</p>

Table 3-5 Options for General Security Parameters

Option	Description
<code>-Dweblogic.management. anonymousAdminLookupEnabled=true</code>	<p>Enables you to retrieve an MBeanHome interface without specifying user credentials. The MBeanHome interface is part of the WebLogic Server JMX API.</p> <p>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.</p> <p>This startup option overrides the Anonymous Admin Lookup Enabled setting on the Security —General tab in the Administration Console.</p> <p>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 "Security Roles" in the <i>Securing WebLogic Resources</i> guide.</p>
<code>-Dweblogic.security. identityAssertionTTL=seconds</code>	<p>Configures the number of seconds that the Identity Assertion cache stores a Subject.</p> <p>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 <code><run-as></code> 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.</p> <p>By default, Subjects remain in the cache for 300 seconds, which is also the maximum allowed value. Setting the value to 0 disables the cache.</p> <p>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.</p>

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 3-27.

Message Output and Logging

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

Table 3-6 Options for Configuring Message Output

Option	Description
<code>-Dweblogic.Stdout="filename"</code>	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 "Redirecting System.out and System.err to a File" in the <i>Administration Console Online Help</i> .
<code>-Dweblogic.Stderr="filename"</code>	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 System.out and System.err to a File" in the <i>Administration Console Online Help</i> .
<code>-Dweblogic. AdministrationMBeanAuditingEnabled= {true false}</code>	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 “Configuration Auditing” in the <i>Administration Console Online Help</i> .

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 3-27.

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 3-27](#).

Other Server Configuration Options

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

Table 3-7 Options for Configuring Server Attributes

Option	Description
<code>-Dweblogic.Name=servername</code>	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 <code>config.xml</code> file.
<code>-Dweblogic.ProductionModeEnabled={true false}</code>	<p>Determines whether a server starts in production mode.</p> <p>A <code>true</code> value prevents a WebLogic Server from automatically deploying and updating applications that are in the <code>domain_name/applications</code> directory.</p> <p>If you do not specify this option, the assumed value is <code>false</code>.</p> <p>For information on the differences between development mode and production mode, see “Differences Between Configuration Startup Modes.”</p>

Table 3-7 Options for Configuring Server Attributes

Option	Description
<code>-Dweblogic.management.startupMode=STANDBY</code>	<p>Starts a server and places it in the STANDBY state. To use this startup argument, the domain must be configured to use the domain-wide administration port.</p> <p>For information about administration ports, see "Enabling the Domain-Wide Administration Port" in the <i>Administration Console Online Help</i>.</p> <p>This startup option overrides any startup mode setting in the <code>config.xml</code> file. The override applies to the current server instantiation; it does not modify the value in the <code>config.xml</code> file. Use the Administration Console, the <code>weblogic.Admin</code> command, or WLST to modify the <code>config.xml</code> file.</p> <p>If you do not specify this value (either on the command line or in <code>config.xml</code>), the default is to start in the RUNNING state.</p>
<code>-Dweblogic.apache.xerces.maxentityrefs=numerical-value</code>	<p>Limits the number of entities in an XML document that the WebLogic XML parser resolves.</p> <p>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.</p>
<code>-Dweblogic.jsp.windows.caseSensitive=true</code>	<p>Causes the JSP compiler on Windows systems to preserve case when it creates output files names.</p> <p>See "Using the WebLogic JSP Compiler" in <i>Developing Web Applications, Servlets, and JSPs for WebLogic Server</i>.</p>

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 3-27.

Clusters

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

Table 3-8 Options for Configuring Cluster Attributes

Option	Description
<code>-Dweblogic.cluster.multicastAddress</code>	<p>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 <code>config.xml</code> file. Use this option to override the value in <code>config.xml</code>.</p> <p>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 3-27.</p> <p>Regardless of how you set the Multicast Address, all servers in a cluster must communicate at the same Multicast Address.</p>

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

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

1. In a command shell, set up the required environment variables by running the following script:
`WL_HOME\server\bin\setWLSEnv.cmd` (on Windows)
`WL_HOME/server/bin/setWLSEnv.sh` (on UNIX)
 where `WL_HOME` is the directory in which you installed the WebLogic Server software.
2. In the command shell, change to the directory that contains your domain's `config.xml` file. For example, change to the `WL_HOME\samples\domains\medrec` directory.
3. To start an Administration Server, enter the following command:
`java -Dweblogic.Name=servername weblogic.Server`
 where `servername` is the name of a server configuration that already exists in the `config.xml` file.

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

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

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

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

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

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

Using the weblogic.Server Command Line to Create a Domain

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

As described in “[Default Behavior](#)” on page 3-3, 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=7701
weblogic.Server
```

After you enter this command, WebLogic Server asks if you want to create a new `config.xml` file. If you enter `y`, it asks you to confirm the password. Then it instantiates a domain named `SimpleDomain`. The domain's Administration Server is configured as follows:

- The name of the Administration Server is `SimpleServer`.
- The domain's security realm defines one administrative user, `weblogic`, with a password of `weblogic`.
- For the listen address of the Administration Server, you can use `localhost`, the IP address of the host computer, or the DNS name of the host computer. For more information about setting the listen address, see "[Configuring the Listen Address](#)" in the *Administration Console Online Help*.
- The Administration Server listens on port 7701.

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

- `config.xml`
- `DefaultAuthenticatorInit.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*.
- `startmydomain.cmd` and `startmydomain.sh`, that you can use to start subsequent instantiations of the server.

Verifying Attribute Values That Are Set on the Command Line

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 the `weblogic.Admin` utility as follows:

```
java weblogic.Admin -url url-for-server-instance -username username
-password password GET -type MBean-nameConfig -property attribute-name
```

For example, to determine the multicast address that a cluster member is using, enter the following command, where `MRMachine1:7041` is the listen address and port of the cluster member:

```
java weblogic.Admin -url MRMachine1:7041 -username weblogic -password  
weblogic GET -pretty -type ClusterConfig -property MulticastAddress
```

To determine the severity level of messages that the example `MedRecServer` prints to standard out, enter the following command:

```
java weblogic.Admin -url localhost:7001 -username weblogic -password  
weblogic GET -pretty -type ServerConfig -property StdoutSeverityLevel
```

For more information about Configuration MBeans, see "[Understanding WebLogic Server MBeans](#)" in *Developing Manageable Applications with JMX*. For more information on using the `weblogic.Admin` utility, see "[weblogic.Admin Command-Line Reference \(Deprecated\)](#)" on [page 1-1](#).

BETA

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, 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 information from all Managed Servers within a domain.

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

- Retrieve the value of WebLogic Server attributes that are exposed as managed objects in the WebLogic Server Management Information Base (MIB).
- Generate and receive WebLogic Server traps.

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

- [“Required Environment and Syntax for the SNMP Command-Line Interface”](#) on page 4-2
- [“Commands for Retrieving the Value of WebLogic Server Attributes”](#) on page 4-4
- [“Commands for Testing Traps”](#) on page 4-9

For more information about using SNMP with WebLogic Server, refer to the [WebLogic SNMP Management Guide](#).

Required Environment and Syntax for the SNMP Command-Line Interface

Before you use the WebLogic SNMP agent command-line interface, set up your environment and note the command syntax information described in the following sections.

Environment

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 [WebLogic Server Installation Guide](#).
2. Enable the WebLogic SNMP agent, as described in "Configuring SNMP and WebLogic Server" in the *Administration Console Online Help*.

Note: The `snmpv1trap` and `snmptrapd` commands do not require the SNMP agent to be enabled.

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.

Common Arguments

All WebLogic SNMP agent commands take the following form:

`java command-name arguments`

[Table 4-1](#) describes arguments that are common to most WebLogic SNMP agent commands.

Table 4-1 Common Command Line Arguments

Argument	Definition
-d	Includes debugging information and packet dumps in the command output.
-v {v1 v2}	<p>Specifies whether to use SNMPv1 or SNMPv2 to communicate with the SNMP agent.</p> <p>You must specify the same SNMP version that you set in the Trap Version field when you configured the SNMP agent (as described in "Enabling and Configuring the WebLogic SNMP Agent" in the <i>Administration Console Online Help</i>)</p> <p>If you do not specify a value, the command assumes -v v1.</p>
-c <i>snmpCommunity</i> [@ <i>server_name</i> @ <i>domain_name</i>]	<p>Specifies the community name that the WebLogic SNMP agent uses to secure SNMP data and specifies the server instance that hosts the objects with which you want to interact.</p> <p>To request the value of an object on the Administration Server, specify: <i>snmpCommunity</i></p> <p>where <i>snmpCommunity</i> is the SNMP community name that you set in the Community Prefix field when you configured the SNMP agent (as described in "Enabling and Configuring the WebLogic SNMP Agent" in the <i>Administration Console Online Help</i>).</p> <p>To request the value of an object on a Managed Server, specify: <i>snmpCommunity@server_name</i></p> <p>where <i>server_name</i> is the name of the Managed Server.</p> <p>To request the value of an object for all server instances in a domain, specify a community string with the following form: <i>snmpCommunity@domain_name</i></p> <p>If you do not specify a value, the command assumes -c public, which attempts to retrieve the value of an object that is on the Administration Server.</p>
-p <i>snmpPort</i>	<p>Specifies the port number on which the WebLogic SNMP agent listens for requests.</p> <p>If you do not specify a value, the command assumes -p 161.</p>
-t <i>timeout</i>	<p>Specifies the number of milliseconds the command waits to successfully connect to the SNMP agent.</p> <p>If you do not specify a value, the command assumes -t 5000.</p>

Table 4-1 Common Command Line Arguments

Argument	Definition
<i>-r retries</i>	Specifies 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.
<i>host</i>	Specifies 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 the Value of WebLogic Server Attributes

Table 4-2 is an overview of commands that retrieve the value of WebLogic Server MBean attributes that are exposed in the WebLogic Server MIB.

Table 4-2 Overview of Commands for Retrieving the Value of WebLogic Server Attributes

Command	Description
<i>snmpwalk</i>	Returns a recursive list of all managed objects that are below a specified node in the MIB tree. See “ snmpwalk ” on page 4-4.
<i>snmpgetnext</i>	Returns a description of the managed object that immediately follows an OID that you specify. See “ snmpgetnext ” on page 4-6.
<i>snmpget</i>	Returns a description of managed objects that correspond to one or more object-instance OIDs. See “ snmpget ” on page 4-8.

snmpwalk

Returns a recursive list of all managed objects that are below a specified node in the MIB tree.

If you specify the OID for an object type, the command returns a list of all instances of that type along with all instances of any child object types.

For example, if you specify the OID for an object type that corresponds to an MBean, this command returns a description of all instances of the MBean **and** all instances of the attributes within the MBeans.

To see the WebLogic Server MIB tree, refer to the [WebLogic Server SNMP MIB Reference](#). For more information about the structure of the MIB and its object identifiers (OIDs), refer to “[Object Identifiers](#)” in *WebLogic SNMP Management Guide*.

Syntax

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

Argument	Definition
<i>OID</i>	Specifies the object ID of the node from which you want to retrieve a recursive list of object values. 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.

For information about the command arguments that are not listed in the above table, refer to [Table 4-1, “Common Command Line Arguments,”](#) on page 4-3.

Example

The following example retrieves the name of all applications that have been deployed on the Administration Server. The OID in the example command is for the `applicationRuntimeName` object type, which represents the `Name` attribute of the `applicationRuntime` MBean.

```
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 type.

```
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. The OID specified in the example command is the numerical value that the WebLogic Server MIB assigns to the `applicationRuntimeName` object type.

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

The following example returns all attributes of the `ServerRuntimeMBean` instance that is hosted on a Managed Server named `MS1`. Note that the OID `.1.3.6.1.4.1.140.625.360` refers to the `serverRuntimeTable` object in the WebLogic MIB.

```
java snmpwalk -c public@MS1 localhost .1.3.6.1.4.1.140.625.360
```

snmpgetnext

Returns a description of the managed object that immediately follows one or more OIDs that you specify.

Instead of the recursive listing that the `snmpwalk` command provides, this command returns the description of only the one managed object 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.

If you specify an object type, this command returns the first instance of the object type, regardless of how many instances of the type exist.

To see the WebLogic Server MIB tree, refer to the [WebLogic Server SNMP MIB Reference](#). For information about the structure of the MIB and its object identifiers (OIDs), refer to “[Object Identifiers](#)” in *WebLogic SNMP Management Guide*.

Syntax

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

Argument	Definition
<i>OID</i> [<i>OID</i>] ...	<p>Specifies one or more object IDs. You can specify an OID for an object type or an object instance.</p> <p>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.</p>

For information about the command arguments that are not listed in the above table, refer to [Table 4-1, “Common Command Line Arguments,” on page 4-3](#).

Example

The following example retrieves the name of an application that has been deployed on the Administration Server. The OID in the example command is for the `applicationRuntimeName` object type, which represents the `Name` attribute of the `applicationRuntime MBean`.

```
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 initial `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 type, which represents the `Name` attribute of the `ApplicationRuntime MBean`, and `jdbcConnectionPoolRuntimeName`, which represents the `Name` attribute of the `JDBCConnectionPoolRuntimeMBean`.

```
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 object types.

Syntax

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


Argument	Definition
<i>object-instance-OID</i> [<i>object-instance-OID</i>]...	The object ID of an object instance. This command does not accept OIDs for object types. 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 value of the `serverRuntimeState` and `serverRuntimeListenPort` attribute instances for the **Administration Server**.

```
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 4-3](#) is an overview of commands that generate and receive traps for testing purposes.

Table 4-3 Overview of Commands for Retrieving Information about WebLogic Server

Command	Description
<code>snmpv1trap</code>	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 4-10 .
<code>snmptrapd</code>	Starts a daemon that receives traps and prints information about the trap. See “snmptrapd” on page 4-13 .

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. For more information about the trap daemon, refer to [“snmptrapd” on page 4-13](#).

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
<code>-c snmpCommunity</code>	Specifies a password (community name) that secures the data in the trap. If you do not specify a value, the command assumes <code>-c public</code> .
<code>-p TrapDestinationPort</code>	Specifies the port number on which the SNMP manager or trap daemon is listening. If you do not specify a value, the command assumes <code>-p 162</code> .
<code>TrapDestinationHost</code>	Specifies the DNS name or IP address of the computer that hosts the SNMP manager or trap daemon.
<code>.1.3.6.1.4.140.625</code>	Specifies the value of the trap's <code>enterprise</code> field, which contains the beginning portion of the OID for all WebLogic Server traps.
<code>agent-addr</code>	Specifies the value of the trap's <code>agent address</code> field. This field is intended to indicate the computer on which the trap was generated. When using the <code>snmpv1trap</code> command to generate a trap, you can specify any valid DNS name or IP address.
<code>generic-trap</code>	Specifies the value of the trap's <code>generic trap type</code> field. For a list of valid values, refer to “ Format of WebLogic Trap Notifications ” in the <i>WebLogic SNMP Management Guide</i> .
<code>specific-trap</code>	Specifies the value of the trap's <code>specific trap type</code> field. For a list of valid values, refer to “ Format of WebLogic Trap Notifications ” in the <i>WebLogic SNMP Management Guide</i> .

Argument	Definition
<i>timestamp</i>	<p>Specifies the value of the trap's timestamp field.</p> <p>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.</p> <p>When using the <code>snmpv1trap</code> command to generate a trap, any number of seconds is sufficient.</p>
OID { INTEGER STRING GAUGE TIMETICKS OPAQUE IPADDRESS COUNTER } <i>value</i>	<p>(Optional) Specifies the value of the trap's variable bindings field, which consists of name/value pairs that further describe the trap notification.</p> <p>For each name/value pair, specify an OID, a value type, and a value.</p> <p>For example, a log message trap includes a <code>trapTime</code> 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 <code>trapTime</code> variable binding, the <code>STRING</code> keyword, and a string that represents the time:</p> <pre>.1.3.6.1.4.1.140.625.100.5 STRING "2:00 pm"</pre>

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

```

snmptrapd

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

Syntax

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

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

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.

Example: Sending Traps to the Trap Daemon

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:
```

Index

A

- Administration commands, overview 1-20, 1-39, 1-52, 4-4, 4-10
- Administration Console
 - specifying private key password for use with SSL 3-16
- Administration Server
 - specifying classpath when starting 3-2
 - starting from command line 3-1
- appc 2-30

C

- CANCEL_SHUTDOWN, WebLogic Server command 1-20
- classpath
 - specifying when starting WebLogic Server 3-2
- Command-line interface
 - administration commands overview 1-20, 1-39, 1-52, 4-4, 4-10
 - command syntax and arguments 1-2, 4-2
 - Mbean management commands overview 1-67, 1-85
- CONNECT, WebLogic Server command 1-21, 1-23
- Connection Pool Administration commands, overview 1-52
- CREATE, WebLogic Server command 1-68
- CREATE_POOL, WebLogic Server command 1-53
- Creating Mbeans, CREATE command 1-68

D

- DDInit 2-11
- DELETE, WebLogic Server command 1-70
- Deleting Mbeans, DELETE command 1-70
- DESTROY_POOL, WebLogic Server command 1-57
- DISABLE_POOL, WebLogic Server command 1-59

E

- ENABLE_POOL, WebLogic Server command 1-61
- EXISTS_POOL, WebLogic Server command 1-65

G

- GET, WebLogic Server command 1-72
- Getting help for a WebLogic Server command 1-43
- Getting Mbean information, GET command 1-72

H

- HELP, WebLogic Server command 1-43
- host name verification
 - default configuration 1-6
 - using 1-6
- Host Name Verifier
 - disabling at start-up 3-15
 - specifying a custom 3-15

- I**
INVOKE, WebLogic Server command 1-74
- J**
Java heap memory
 specifying minimum and maximum 3-6
JHTML 2-19
jhtml2jsp 2-19
JNDI naming tree
 list node bindings 1-44
- L**
LICENSES, WebLogic Server command 1-44
LIST, WebLogic Server command 1-44
Listening ports, verify 1-29
LOCK, WebLogic Server command 1-28
- M**
Mbean management commands, overview 1-67, 1-85
- P**
PING, WebLogic Server command 1-29
- R**
RESET_POOL, WebLogic Server command 1-63
Resetting connection pools, RESET_POOL command 1-63
- S**
Secure Sockets Layer (SSL)
 host name verification 1-6
 server name
 specifying at startup 3-23
SERVERLOG, WebLogic Server command 1-47
SET, WebLogic Server command 1-79
Setting attribute values, SET command 1-79
SHUTDOWN, WebLogic Server command 1-30
SSL
 specifying private key password at server startup 3-16
SSL session caching
 indicating 3-16
system home directory, WebLogic
 specifying at startup 3-7
- T**
THREAD_DUMP, WebLogic Server command 1-49
Threads, view running 1-49
- U**
UNLOCK, WebLogic Server command 1-39
- V**
Verify WebLogic Server listening ports 1-29
VERSION, WebLogic Server command 1-50
Viewing server log files, SERVERLOG command 1-47
- W**
WebLogic Server
 licenses, viewing 1-44
 specifying user name of at startup 3-18
WebLogic Server commands
 administration commands overview 1-20, 1-39, 1-52, 4-4, 4-10
 CANCEL_SHUTDOWN 1-20
 CONNECT 1-21, 1-23
 connection pool commands overview 1-52
 CREATE 1-68
 CREATE_POOL 1-53
 DELETE 1-70

DESTROY_POOL 1-57
DISABLE_POOL 1-59
ENABLE_POOL 1-61
EXISTS_POOL 1-65
GET 1-72
HELP 1-43
INVOKE 1-74
LICENSES 1-44
LIST 1-44
LOCK 1-28
Mbean management commands overview
 1-67, 1-85
PING 1-29
RESET_POOL 1-63
SERVERLOG 1-47
SET 1-79
SHUTDOWN 1-30
 syntax and arguments 1-2, 4-2
THREAD_DUMP 1-49
UNLOCK 1-39
VERSION 1-50
WebLogicObjectName
 defined 1-76

BETA

BETA