Oracle® Fusion Middleware

Command Reference for Oracle WebLogic Server



26c (26.1.0.0.0)
G56915-01
November 2026

ORACLE

Oracle Fusion Middleware Command Reference for Oracle WebLogic Server, 26c (26.1.0.0.0)

G56915-01

Copyright © 2007, 2026, Oracle and/or its affiliates.

Primary Author: Oracle Corporation

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Contents

Preface

Audience i
Documentation Accessibility i
Diversity and Inclusion i
Related Documentation i
Conventions ii

1 Using the Oracle WebLogic Server Java Utilities

appc 1
AppletArchiver 1
    Syntax 1
BuildXMLGen 2
CertGen 2
    Syntax 2
    Example 4
ClientDeployer 5
clientgen 6
dbping 6
    Creating a DB2 Package with dbping 6
    Syntax 7
    Examples 8
DemoCertGen 9
Deployer 9
DeploymentPersistentTool 10
der2pem 11
    Syntax 11
    Example 11
Derby 11
domainKeystores.sh 12
ejbc (deprecated) 13
encrypt 14
    Syntax 14
    Examples 15

Copyright © 2007, 2026, Oracle and/or its affiliates.

getProperty 15
Syntax 15
Example 15
host2ior 15
Syntax 15
ImportPrivateKey 15
Syntax 16
Example 17
jspc (deprecated) 17
logToZip 18
Syntax 18
Examples 18
MBean Commands 18
MulticastTest 19
Syntax 19
Example 20
myip 20
Syntax 20
Example 21
pem2der 21
Syntax 21
Example 21
rmic 21
Schema 21
Syntax 22
Example 22
SearchAndBuild 22
Example 22
system 23
Syntax 23
Example 23
ValidateCertChain 23
verboseToZip 24
Syntax 24
WebLogicMBeanMaker 24
Syntax 24
wlappc 25
wlcompile 25
wlconfig 25
wldeploy 25
wlpackage 25

Copyright © 2007, 2026, Oracle and/or its affiliates.

wlserver 25

2 weblogic.Server Command-Line Reference

Required Environment and Syntax for weblogic.Server 1
Environment 1
Modifying the Classpath 1
Syntax 2
Default Behavior 2
weblogic.Server Configuration Options 3
JVM Parameters 4
Location of Configuration Data 5
Example 5
Options that Override a Server's Configuration 6
Server Communication 6
SSL 8
HTTP Strict Transport Security 14
Security 15
Message Output and Logging 21
Clusters 22
Deployment 23
Other Server Configuration Options 23
Using the weblogic.Server Command Line to Start a Server Instance 26
Using the weblogic.Server Command Line to Create a Domain 27
Verifying Attribute Values That Are Set on the Command Line 28

Copyright © 2007, 2026, Oracle and/or its affiliates.

Preface

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

Audience

This document is written for system administrators and application developers deploying e-commerce applications using the Jakarta EE platform. It is assumed that readers are familiar with Web technologies and the operating system and platform where Oracle WebLogic Server is installed.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documentation

- Using Ant Tasks to Configure and Use a WebLogic Server Domain in *Developing Applications with Oracle WebLogic Server*
- *Understanding the WebLogic Scripting Tool*
- *Administering Server Environments for Oracle WebLogic Server*
- *Oracle WebLogic Remote Console Online Help*

Copyright © 2007, 2026, Oracle and/or its affiliates.

New and Changed WebLogic Server Features

For a comprehensive listing of the new WebLogic Server features introduced in this release, see *What's New in Oracle WebLogic Server*.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for text entered by users.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that user enter.

Copyright © 2007, 2026, Oracle and/or its affiliates.

1

Using the Oracle WebLogic Server Java Utilities

Oracle WebLogic Server provides a number of Java utilities and Ant tasks for performing administrative and programming tasks, installing and configuring the WebLogic Server environment, building and deploying applications, generating certificates for development environments, providing convenient shortcuts, and more.

To use these utilities and tasks, you must set your `CLASSPATH` correctly. For more information, see Modifying the Classpath. The command-line syntax is specified for all utilities and, for some, examples are provided.

The Apache Web site provides other useful Ant tasks as well, including tasks for packaging EAR, WAR, and JAR files. For more information, see `https://ant.apache.org/manual/`.

appc

The `appc` compiler generates and compiles the classes needed to deploy EJBs and JSPs to Oracle WebLogic Server. It also validates the deployment descriptors for compliance with the current specifications at both the individual module level and the application level. See appc Reference in *Developing Jakarta Enterprise Beans Using Deployment Descriptors*.

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
```

Table 1-1 describes the arguments passed to the `AppletArchiver` utility.

Table 1-1 AppletArchiver Arguments

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

Copyright © 2007, 2026, Oracle and/or its affiliates.

BuildXMLGen

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

CertGen

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

The `CertGen` utility generates certificates with the following attributes by default:

- 2048-bit public key.
- SHA256 message digest algorithm.
- Subject Key Identifier extension.
- Authority Key Identifier extension (if the CA certificate contains a Subject Key ID.)

Syntax

```
$ java utils.CertGen
      -certfile <cert_file> -keyfile <private_key_file>
      -keyfilepass <private_key_password>
     [-cacert <ca_cert_file>][-cakey <ca_key_file>]
     [-cakeypass <ca_key_password>]
     [-selfsigned][-strength <key_strength>]
     [-digestalgorithm <message digest algorithm such as MD5, SHA1, or
SHA256>]
     [-e <email_address>][-cn <common_name>]
     [-ou <org_unit>][-o <organization>]
     [-l <locality>][-s <state>][-c <country_code>]
     [-keyusage [digitalSignature,nonRepudiation,keyEncipherment,
       dataEncipherment,keyAgreement,keyCertSign,
       cRLSign,encipherOnly,decipherOnly]]
     [-keyusagecritical true|false]
     [-subjectkeyid <subject_key_identifier>]
     [-subjectkeyidformat UTF-8|BASE64]
     [-nosanhostdns]
     [-a DNS:<hostname>,DNS:<localhost>,IP:<ip address>,IP:<127.0.0.1>]
     [-domain <DOMAIN_HOME>]
     [-validuntil "YYYY-MM-DD"]
     [-help]
```

Table 1-2 describes the arguments that are passed to the `CertGen` utility.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 1-2 CertGen Arguments

Argument	Definition
`-certfile` *`cert_file`* `-keyfile` *`private_key_file`*	Respectively, the output file names without extensions of the generated public certificate and private key. The appropriate extensions are appended when the `pem` and `der` files are created.
`-keyfilepass` *`private_key_password`*	The password for the generated private key.
`-cacert` *`ca_cert_file`* `-cakey` *`ca_key_file`* `-cakeypass` *`ca_key_password`*	Respectively, the public certificate, private key file, and private key password of the CA that will be used as the issuer of the generated certificate. If one or more of these options are not specified, the relevant demonstration CA files will be used: `CertGenCA.der` and `CertGenCAKey.der`. The `CertGen` utility first looks in the current working directory, then in the *`WL_HOME`*`/lib` directory.
`-selfsigned`	Generates a self-signed certificate that can be used as a trusted CA certificate. If this argument is specified, the *`ca_cert_filename`*, *`ca_key_filename`*, and *`ca_key_password`* arguments should not be specified.
`-digestalgorithm [message digest algorithm]`	The message digest algorithm used with the signature algorithm to sign the certificate. The default is SHA256. Supported values are MD5, SHA1, SHA256, SHA384, and SHA512.
`-strength` *`key_strength`*	The length (in bits) of the keys to be generated. The default is 2048 bits. The longer the key, the more difficult it is for someone to break the encryption. Generating a certificate with an RSA key length less than 1024 bits may not work in JDK 7u40+. See `http://www.oracle.com/technetwork/java/javase/7u40-relnotes-2004172.html` for additional information.
`-e` *`email_address`*	The email address associated with the generated certificate.
`-cn` *`common_name`*	The name associated with the generated certificate.
`-ou` *`org_unit`*	The name of the organizational unit associated with the generated certificate.
`-o` *`organization`*	The name of the organization associated with the generated certificate.
`-l` *`locality`*	The name of a city or town.
`-s` *`state`*	The name of the state or province in which the organizational unit (`ou`) operates if your organization is in the United States or Canada, respectively. Do not abbreviate.
`-c` *`country_code`*	Two-letter ISO code for your country. The code for the United States is US.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 1-2 (Cont.) CertGen Arguments

Argument	Definition
`-keyusage [digitalSignature, nonRepudiation,keyEncipherment, dataEncipherment,keyAgreement, keyCertSign,cRLSign, encipherOnly,decipherOnly]`	Generate certificate with a key usage extension, and with bits set according to the comma-separated list of bit names. Specify a key usage when you want to restrict the operation for a key that could be used for more than one operation.
`-keyusagecritical true|false`	By default, a key usage extension is marked critical. To generate a certificate with a non-critical extension, use `-keyusagecritical false`.
`-subjectkeyid` *`subject_key_identifier`*	Generates a certificate with the specified subject key identifier.
`-subjectkeyidformat UTF-8|BASE64`	The format of the `subjectkeyid` value; UTF-8 is the default.
`-nosanhostdns`	Disables the use of the fully-qualified DNS name in the SAN extension. By default, without this option, the certificate is generated with the SAN extension containing the fully qualified DNS name.
`-a`	Specify a comma-separated list of additional host names, or IP addresses, or both, in the Subject Alternative Name (SAN) field of the certificate, by using the `-a DNS:<`*`hostname`*`>,IP:<`*`ip address`*`>` option.
`-domain <`*`DOMAIN_HOME`*`>`	Uses the specified domain's demo CA to sign certificates, save the identity and trust stores in the *`DOMAIN_HOME`*`/security` directory and the passphrases in the `DemoCerts.props` file.
`-validuntil "`*`YYYY-MM-DD`*`"`	Specifies how long a certificate or CA is valid for. By default, certificates are valid for 6 months and demo CAs are valid for 5 years.

Example

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

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

```
$ java utils.CertGen -keyfilepass mykeypass
-certfile testcert -keyfile testkey
Generating a certificate with common name machine-name and key strength 2048
issued by CA with certificate from CertGenCA.der file and key from CertGenCAKey.der file
```


Copyright © 2007, 2026, Oracle and/or its affiliates.

ClientDeployer

Use `weblogic.ClientDeployer` to extract the client-side JAR file from a Jakarta EE EAR file, creating a deployable JAR file.

Run the `weblogic.ClientDeployer` class on the Java command line with the following syntax:

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

Table 1-3 Syntax for ClientDeployer

Argument	Definition
`ear-file`	An expanded directory (or Java archive file with a `.ear` extension) that contains one or more client application JAR files.
`client`	The name of the client.

For example:

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

In the preceding example, `app.ear` is the EAR file that contains a Jakarta EE client packaged in `myclient.jar`.

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

Run the `weblogic.j2eeclient.Main` class on the Java command line with the following syntax:

```
java weblogic.j2eeclient.Main [options] [clientjar] URL [client-args]
```


Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 1-4 Syntax for weblogic.j2eeclient.Main

Argument	Definition
options	Optional. Options can include: • `-clientName` *client-jar-name*: The name of the client jar to be invoked. Use this option when passing an exploded EAR. • `-serverApplicationName` *server-application-name*: The name of the corresponding application on the server. • `-userConfigFile <user-config-file>`: The filename of the user configuration file that contains the username and password to use when connecting to the server. • `-userKeyFile <user-key-file>`: The filename of the user key file to be used when decrypting the username and password in the user configuration file. For more information on user configuration files, see storeUserConfig in *WLST Command Reference for Oracle WebLogic Server*.
clientjar	Optional. The client jar or an exploded `ear` containing the client jar.
URL	The address of the administration server.
client-args	Optional. Additional arguments for the client.

For example:

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

clientgen

Use `clientgen` to generate the client-side artifacts, needed to invoke a Web Service.

See Ant Task Reference in *WebLogic Web Services Reference for Oracle WebLogic Server*.

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. To install a driver, see the documentation from your driver vendor. Also see Using Third-Party Drivers with WebLogic Server in *Developing JDBC Applications for Oracle WebLogic Server*.

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

Copyright © 2007, 2026, Oracle and/or its affiliates.

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 Using DataDirect Documentation in *Developing JDBC Applications for Oracle WebLogic Server*.) You can use the `-d` option with dynamic sections set at `200` to forcibly recreate a default package on the DB2 server.

> **Note**
>
> 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
```

Table 1-5 describes the arguments that are passed to the `dbping` command-line utility.

Table 1-5 dbping Arguments

Argument	Definition
DBMS	Varies by DBMS and JDBC driver: `DB2B`—WebLogic Type 4 JDBC Driver for DB2 `DERBY`—Derby network client driver `JCONN2`—Sybase JConnect (JDBC 2.0) driver `JCONN3`—Sybase JConnect (JDBC 2.0) driver `JCONNECT`—Sybase JConnect driver `INFORMIXB`—WebLogic Type 4 JDBC Driver for Informix `MSSQLSERVERB`—WebLogic Type 4 JDBC Driver for Microsoft SQL Server `MYSQL`— MySQL's Type 4 Driver `ORACLEB`—WebLogic Type 4 JDBC Driver for Oracle `ORACLE_THIN`—Oracle Thin Driver `POINTBASE`—PointBase Universal Driver `SYBASEB`—WebLogic Type 4 JDBC Driver for Sybase
`[-d` *`dynamicSections`*`]`	Specifies the number of dynamic sections to create in the DB2 package. This option is for use with the WebLogic Type 4 JDBC Driver for DB2 only. If the `-d` option is specified, the driver automatically sets CreateDefaultPackage=true and ReplacePackage=true on the connection and creates a DB2 package with the number of dynamic sections specified.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 1-5 (Cont.) dbping Arguments

Argument	Definition
user	Valid database username for login. Use the same values you use with `isql`, `sqlplus`, or other SQL command-line tools. For DB2 with the `-d` option, the user must have CREATE PACKAGE privileges on the database.
password	Valid database password for the user. Use the same values you use with `isql` or `sqlplus`.
DB	Name and location of the database. Use the following format, depending on which JDBC driver you use: `DB2B`—Host:Port/DBName `DERBY`—Host:Port/DBName `JCONN2`—Host:Port/DBName `JCONN3`—Host:Port/DBName `JCONNECT`—Host:Port/DBName `INFORMIXB`—Host:Port/DBName/InformixServer `MSSQLSERVERB`—Host:Port[/DBName] `MYSQL`—[Host][:Port]/[DBName] `ORACLEB`—Host:Port/DBName `ORACLE_THIN`—Host:Port:DBName `POINTBASE`—Host[:Port]/DBName `SYBASEB`—Host:Port/DBName Where: • *Host* is the name of the machine hosting the DBMS. • *Port* is port on the database host where the DBMS is listening for connections. • *DBName* is the name of a database on the DBMS. • *InformixServer* is an Informix-specific environment variable that identifies the Informix DBMS server.

Examples

The following is an example using the Oracle Thin Driver.

```
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);
```


Copyright © 2007, 2026, Oracle and/or its affiliates.

The following is an example using the Derby driver. Derby is an open source relational database management system bundled with WebLogic Server for use by the sample applications and code examples as a demonstration database.

```
$ java utils.dbping DERBY examples examples localhost:1527/demo
**** Success!!! ****
You can connect to the database in your app using:

  java.util.Properties props = new java.util.Properties();
  props.put("user", "examples");
  props.put("password", "examples");
  java.sql.Driver d =
    Class.forName("org.apache.derby.jdbc.ClientDriver").newInstance();
  java.sql.Connection conn =
    Driver.connect("jdbc:derby://localhost:1527/demo", props);
```

DemoCertGen

The DemoCertGen utility configures keystore resources for demonstration or testing purposes. Do not use it in a production environment.

The `DemoCertGen` utility combines functionality from the `CertGen`, Java `keytool`, and `ImportPrivateKey` utilities to configure demonstration keystores for each domain.

1. The `CertGen` utility generates a self-signed demonstration certificate authority (CA) for the domain.
2. The Java `keytool` utility imports the demo CA to the demo trust keystore, `DemoTrust.p12`, under the `CertGenCA_<actual domain name>` alias.
3. The `CertGen` utility then generates a demo identity certificate signed with the newly created demo CA.
4. The `ImportPrivateKey` utility imports the demo certificate and private key to the demo identity keystore, `DemoIdentity.p12`, under the `demoidentity` alias.

Syntax

```
$ java utils.DemoCertGen -domain DOMAIN_HOME [-genIDOnly] [-cn common_name]
```

Table 1-6 DemoCertGen Arguments

Argument	Definition
`-domain DOMAIN_HOME`	Generates and imports a demonstration certificate, signed with the demo CA.
`-genIDOnly`	Generates the demo identity store only.
`-cn common_name`	The name associated with the generated certificate.

Deployer

Using the weblogic.Deployer tool, you can deploy Jakarta EE applications and components to WebLogic Servers in a command-line or scripting environment. For detailed information on

Copyright © 2007, 2026, Oracle and/or its affiliates.

using this tool, see weblogic.Deployer Command-Line Reference in *Deploying Applications to Oracle WebLogic Server*.
The `weblogic.Deployer` utility replaces the `weblogic.deploy` utility, which has been deprecated.

DeploymentPersistentTool

The `DeploymentPersistentTool` utility reads the application's runtime state from the persistent store.

Syntax

```
$ java weblogic.deploy.utils.DeploymentPersistentTool [-list] [-remove] [-dir
directory] [-store store-name] [-appexp application-id-regular-expression] [-
nofilecheck]
```

Table 1-7 DeploymentPersistentTool Arguments

Argument	Definition
`-list`	Lists the application's runtime states.
`-remove`	Removes the application runtime states.
`-dir` *`directory`*	Specify the name of the directory containing the store. File path can be either absolute or relative.
`-store` *`store-name`*	Specify the name of the persistence store to open.
`-appexp` *`application-id-regular-expression`*	Specify which application's names to print using a user specified regular expression as a filter. By default, this is set to `.*` and matches all application names.
`-nofilecheck`	When set, the DeploymentPersistentTool will not check for the existence of a file prior to running.

Examples

This example runs the `DeploymentPersistentTool` utility on the Administration Server:

```
$ java weblogic.deploy.utils.DeploymentPersistentTool -list -dir $
{adminServer.dir}/servers/adminServer/data/store/default -store
_WLS_ADMINSERVER
```

This example runs the `DeploymentPersistentTool` utility on different Managed Servers:

```
$ java weblogic.deploy.utils.DeploymentPersistentTool -list -dir $
{adminServer.dir}/servers/server1/data/store/default -store _WLS_SERVER1

$ java weblogic.deploy.utils.DeploymentPersistentTool -list -dir $
{adminServer.dir}/servers/server2/data/store/default -store _WLS_SERVER2
```


Copyright © 2007, 2026, Oracle and/or its affiliates.

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]
```

Table 1-8 describes the arguments that are passed to the `der2pem` utility.

Table 1-8 der2pem Arguments

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

Example

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

Derby

Derby is an open source relational database management system based on Java, JDBC, and SQL standards. It is bundled with WebLogic Server for use by the sample applications and code examples as a demonstration database.See http://db.apache.org/derby.

Copyright © 2007, 2026, Oracle and/or its affiliates.

domainKeystores.sh

The `domainKeystores.sh` utility manages the keystore resources made available through the Certificate Management Service. You should *only* use this command when the domain is offline.

For more information, see Managing Certificates with the Certificate Management Service in *Administering Security for Oracle WebLogic Server*.

The `domainKeystores.sh` utility wraps the Java `keytool` utility to provide offline access to the Domain Keystores that are administered by the Certificate Management Service.

If you need to manage Domain Keystores in a running domain, use WebLogic Remote Console or the appropriate MBeans instead. See Configure the Certificate Management Service in *Oracle WebLogic Remote Console Online Help* or DomainKeystoresDomainRuntimeMBean and DomainKeystoresServerRuntimeMBean in *MBean Reference for Oracle WebLogic Server*.

You must have access to the file system of the target domain and its keystore files.

Syntax

```
domainKeystores.sh -h
domainKeystores.sh -printcert -file fileName [-v]
domainKeystores.sh -exporttrust -file fileName [-passphrase passphrase] [-v]
domainKeystores.sh -keystore keystore-name [-server serverName] -list [-v]
domainKeystores.sh -keystore keystore-name [-server serverName] -delete -
alias aliasName [-v]
domainKeystores.sh -keystore keystore-name [-server serverName] -importcert -
alias aliasName -file fileName [-v]
```

Table 1-9 domainKeystores Arguments

Argument	Definition
`-h`	Display command usage.
`-keystore keystoreName`	Enter the keystore name as listed in Keystores in *Administering Security for Oracle WebLogic Server*. You do not need to include the file path or extension. For server keystores, you can also use the `-server` argument to specify the keystore instead.
`-alias alias`	Enter the alias of an entry to import or delete.
`-delete -keystore keystoreName -alias alias`	Delete a specified entry in the selected keystore.
`-exporttrust -file outputFile`	Export the contents of the `domain-trust` and `provisioned-trust` keystores, for use in another keystore. Use with the `-file` option.
`-file`	Specify the certificate file for `-printcert` or `-importcert`, or the destination keystore file for `-exporttrust`.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 1-9 (Cont.) domainKeystores Arguments

Argument	Definition
`-importcert -keystore` *`keystoreName`* `-alias` *`alias`* `-file` *`fileName`*	Import an identity or trust certificate. You can only import into a user-writeable keystore. Use with the `-file` option.
`-list`	List all of the entries in a specified keystore.
`-passphrase`	Specify the passphrase to set for the destination keystore where the trust certificates will be exported to. Use with the `exporttrust` argument only.
`-printcert -file` *`fileName`*	Print the certificate in the file specified by `-file`. Do not use `-keystore` or `-alias` with this command.
`-server` *`serverName`*	Specify the server to which the keystore belongs. Only use this argument with server specific keystores. For example: `-keystore server-trust -server <`*`servername`*`>`.
`-v`	Produce verbose output.

Example

The following is an example of using the `domainKeystores` utility to import a certificate into the `provisioned-identity` keystore.

```
domainKeystores.sh -keystore provisioned-identity -importcert -alias newalias
-v -file CERT_FILE_TO_IMPORT
```

The following is an example of using the `domainKeystores` utility to list all of the certificates within the `provisioned-identity` keystore.

```
domainKeystores.sh -keystore provisioned-identity -list
```

The following is an example of using the `domainKeystores` utility to delete the *KEY_ALIAS* certificate from the `provisioned-identity` keystore.

```
domainKeystores.sh -keystore provisioned-identity -delete -alias KEY_ALIAS
```

ejbc (deprecated)

For each deployment descriptor either specified in the command line or present in a `.jar` file, `ejbc` creates wrapper classes for the corresponding EJBean class. It then runs these through the RMI compiler, which generates a client-side stub and a server-side skeleton. See appc Reference in *Developing Jakarta Enterprise Beans Using Deployment Descriptors*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

encrypt

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

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

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

> **Note**
>
> Oracle recommends running the utility from the Administration Server domain directory or on the machine hosting the Administration Server and specifying a domain root directory.

Syntax

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

Table 1-10 describes the arguments that are passed to the `weblogic.security.Encrypt` utility.

Table 1-10 Encrypt Arguments

Argument	Definition
dirname	Optional. Oracle WebLogic Server domain directory in which the encrypted string will be used. If not specified, the default domain root directory is the current directory (the directory in which the utility is being run).
`weblogic.management.allowPasswordEcho`	Optional. Allows echoing characters entered on the command line. `weblogic.security.Encrypt`expects that no-echo is available; if no-echo is not available, set this property to `true`.
password	Optional. Cleartext string to be encrypted. If omitted from the command line, you will be prompted to enter a password.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Examples

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

```
java weblogic.security.Encrypt xxxxxx
{AES}yWv/i0qhfM4/IvzoghzjHj/xpJUkQPF8OWuSfh0f0Ss=
```

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

```
java -Dweblogic.RootDirectory=./mydomain weblogic.security.Encrypt xxxxxx
{AES}wr86u9Z5DHr+5p7WIbzTDSy4M/sl7EYnX/K5xzcarDQ=
```

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

```
java weblogic.security.Encrypt
Password:
{AES}LIX8hoiStcAhph0PGCpveouw/0UO0lciODuj+TQh/bs=
```

getProperty

The `getProperty` utility displays Java system properties. If you supply one or more property names, it prints each requested name and value. With no arguments, it lists all system properties.

Syntax

```
$ java utils.getProperty [propertyName ...]
```

Example

```
$ java utils.getProperty java.version os.name
java.version=<installed JDK version>
os.name=<operating system name>
```

host2ior

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

Syntax

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

ImportPrivateKey

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

Copyright © 2007, 2026, Oracle and/or its affiliates.

Syntax

```
$ java utils.ImportPrivateKey
      -certfile <cert_file> -keyfile <private_key_file>
     [-keyfilepass <private_key_password>]
      -keystore <keystore> -storepass <storepass> [-storetype <storetype>]
      -alias <alias> [-keypass <keypass>]
     [-domain <DOMAIN_HOME>]
     [-help]
```

Table 1-11 describes the arguments that are passed to the `ImportPrivateKey` utility.

Table 1-11 ImportPrivateKey Arguments

Argument	Definition
`-certfile cert_file`	The name of the certificate associated with the private key.
`-keyfile private_key_file`	The name of the generated private key file.
`-keyfilepass private_key_password`	The password for the private key.
`-keystore keystore`	The name of the keystore file. A new keystore is created if one does not exist.
`-storepass storepass`	The password for the keystore.
`-storetype storetype`	The type (format) of the keystore. The *storetype* argument, which is the same as that used by the `keytool` command, specifies the type of Java keystore. The default store type is the value returned by the active JDK's `KeyStore.getDefaultType()`, which is controlled by the `keystore.type` security property. To avoid depending on the runtime default, specify `-storetype` explicitly (for example, `pkcs12`, `jks`, or another type supplied by a configured security provider).
`-alias alias`	The name that is used for looking up the certificate and private key being imported into the keystore.
`-keypass keypass`	The password of the private key entry being imported into the keystore. If *keypass* is not specified, the first default is *private_key_password*, and the second default is *storepass*.
`-domain <DOMAIN_HOME>`	Use the specified domain's *DOMAIN_HOME*`/security/DemoCerts.props` file to save the passphrases.

> **Note**
>
> If you used `CertGen` to create a private key file protected by a password (`-keyfilepass` *private_key_password*), that password is the one required by `ImportPrivateKey` to extract the key from the key file and insert the key in the newly created keystore (which will contain both the certificate(s) from *cert_file* and the private key from *private_key_file*).

Copyright © 2007, 2026, Oracle and/or its affiliates.

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

> **Note**
>
> By default, the CertGen utility looks for the `CertGenCA.der` and `CertGenCAKey.der` files in the current directory, or in the *`WL_HOME`*`/server/lib` directory, as specified in the `weblogic.home` system property or the CLASSPATH.
>
> Alternatively, you can specify CA files on the command line. If you want to use the default settings, there is no need to specify CA files on the command line.

To generate a certificate:

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

   ```
   $ java utils.CertGen -keyfilepass mykeyfilepass
   -certfile testcert -keyfile testkey
   Generating a certificate with common name machine-name and key strength 2048
   issued by CA with certificate from CertGenCA.der file and key from CertGenCAKey.der
   file
   ```

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

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

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

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

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

   ```
   $ java utils.ImportPrivateKey -keystore mykeystore -storepass password \
       -keyfile testkey.pem -keyfilepass mykeyfilepass \
       -certfile newcerts.pem -alias passalias -storetype jks

   No password was specified for the key entry
   Key file password will be used
   .
   .
   .
   Imported private key testkey.pem and certificate newcerts.pem
   into a new keystore mykeystore of type jks under alias passalias
   ```

jspc (deprecated)

The `jspc` utility is a JSP-specific compiler task. Use appc instead.

Copyright © 2007, 2026, Oracle and/or its affiliates.

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
```

Table 1-12 describes the arguments that are passed to the `logToZip` utility.

Table 1-12 logToZip Arguments

Argument	Definition
logfile	Required. Fully-qualified pathname of the log file.
codebase	Required. Code base for the applet, or "" if there is no code base. By concatenating the code base with the full package name of the applet, you get the full pathname of the applet (relative to the HTTP document root).
zipfile	Required. Name of the `.zip` file to create. The resulting `.zip` file is created in the directory in which you run the program. The path name for the specified file can be relative or absolute. In the examples shown below, a relative path name 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\access applets\classes app3.zip
```

Here `\logs\access` is the log file, `applets\classes` is the codebase relative to the HTTP document root, and `app3.zip` is the output archive.

MBean Commands

Use the MBean commands (`CREATE`, `DELETE`, `GET`, `INVOKE`, and `SET`) to administer MBeans. See Editing Commands in *WLST Command Reference for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

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.

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

For information about setting up multicast, see the configuration documentation for the operating system and hardware of the WebLogic Server host machine. See *Administering Clusters for Oracle WebLogic Server*.

Syntax

```
$ java utils.MulticastTest -n name -a address [-p portnumber]
     [-t timeout] [-s send] [-i interfaceAddress] [-l timeToLive] [-b true|
false] [-h]
```

Table 1-13 describes the arguments that are passed to the `MulticastTest` utility.

Table 1-13 MulticastTest Arguments

Argument	Definition
`-n name`	Required. A name that identifies the sender of the sequenced messages. Use a different name for each test process you start.
`-a address`	The multicast address on which: (a) the sequenced messages should be broadcast; and (b) the servers in the clusters are communicating with each other. (The default is 237.0.0.1.)
`-p portnumber`	Optional. The multicast port on which all the servers in the cluster are communicating. (The multicast port is the same as the listen port set for WebLogic Server, which defaults to 7001 if not set.)

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 1-13 (Cont.) MulticastTest Arguments

Argument	Definition
`-t` *`timeout`*	Optional. Idle timeout, in seconds, if no multicast messages are received. If not set, the default is 600 seconds (10 minutes). If a timeout is exceeded, a positive confirmation of the timeout is sent to `stdout`.
`-s` *`send`*	Optional. Interval, in seconds, between sends. If not set, the default is 2 seconds. A positive confirmation of each message sent out is sent to `stdout`.
`-i` *`interfaceAddress`*	Optional. Address of the network interface to use. If omitted, the default interface is used.
`-l` *`timeToLive`*	Optional. Maximum multicast hop count. The default is 1, which restricts packets to the local subnet.
`-b true|false`	Optional. If true, bind the socket to the multicast address specified by `-a` instead of `IP_ANY`. The default is false.
`-h`	Display command usage.

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
```


Copyright © 2007, 2026, Oracle and/or its affiliates.

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
```

Table 1-14 describes the argument that is passed to the `pem2der` utility.

Table 1-14 pem2der Arguments

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

Example

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

rmic

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

Schema

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

Copyright © 2007, 2026, Oracle and/or its affiliates.

Syntax

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

Table 1-15 describes the arguments that are passed to the `Schema` utility.

Table 1-15 Schema Arguments

Argument	Definition
driverURL	Required. URL for the JDBC driver.
driverClass	Required. Pathname of the JDBC driver class.
`-u` *username*	Optional. Valid username.
`-p` *password*	Optional. Valid password for the user.
`-verbose`	Optional. Prints database error metadata. SQL statements and exception messages are not printed.
SQLfile	Required. Text file with SQL statements.

Example

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

```
$ java utils.Schema
"jdbc:derby://localhost:1527/demo"
"org.apache.derby.jdbc.ClientDriver" -u examples
-p examples examples/utils/ddl/demo.ddl

utils.Schema will use these parameters:
       url: ******
    driver: org.apache.derby.jdbc.ClientDriver
      user: examples
  password: ******
  SQL file: examples/utils/ddl/demo.ddl
```

SearchAndBuild

This Ant task executes `build.xml` files that are included within the `FileSet`. The task assumes that all of the files defined in `FileSet` are valid build files, and executes the `Ant` task of each of them.
Make certain that your `FileSet` filtering is correct. If you include the `build.xml` file that `SearchAndBuildTask` is being called from, you will be stuck in an infinite loop as this task will execute the top level build file—itself—forever.

Example

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


Copyright © 2007, 2026, Oracle and/or its affiliates.

```
<include name="**\build.xml"/>
<exclude name="build.xml"/>
</fileset>
</buildAll>
</target>
</project>
```

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
```

The output reports the active runtime's `java.version`, `java.vendor`, `java.class.path`, `os.name`, `os.arch`, and `os.version` values. Exact values depend on the JDK installation and host operating system.

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.

> **Note**
>
> The `-file` and `-pem` options are deprecated; use `-pkcs12store` or `-jks` instead. The `-pkcs12file` option is also deprecated; use `-pkcs12store` or `-jks` instead.

A complete certificate chain must be used with the utility. The following is the syntax for the `ValidateCertChain` utility:

```
java utils.ValidateCertChain -file pemcertificatefilename (Deprecated, use -pkcs12store
or -jks)
java utils.ValidateCertChain -pem pemcertificatefilename (Deprecated, use -pkcs12store
or -jks)
java utils.ValidateCertChain -pkcs12store pkcs12storefilename
java utils.ValidateCertChain -pkcs12file pkcs12filename password (Deprecated, use -
pkcs12store or -jks)
java utils.ValidateCertChain -jks alias storefilename [storePass]
```

Example of valid certificate chain:

```
java utils.ValidateCertChain -jks mykey mykeystore
Cert[0]: CN=zippy,OU=FORTESTINGONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US
Cert[1]: CN=CertGenCAB,OU=FOR TESTINGONLY,O=MyOrganization,L=MyTown,ST=MyState,C=US
```


Copyright © 2007, 2026, Oracle and/or its affiliates.

```
Certificate chain appears valid
```

Example of invalid certificate chain:

```
java utils.ValidateCertChain -jks mykey mykeystore
Cert[0]: CN=corba1 OU=FOR TESTING ONLY, O=MyOrganization,L=MyTown,ST=MyState,C=US
CA cert not marked with critical BasicConstraint indicating it is a CA
Cert[1]: CN=CACERT,OU=FOR TESTING ONLY, O=MyOrganization,L=MyTown,ST=MyState,C=US

Certificate chain is invalid
```

verboseToZip

The `verboseToZip` utility reads a legacy Java class-load log whose class records begin with `[Loaded ...]` and creates an uncompressed `.zip` file containing the referenced classes. The current utility does not accept the unified class-load records produced by JDK 21, such as `[0.008s][info][class,load] ....`

Syntax

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

Table 1-16 describes the arguments that are passed to `verboseToZip`.

Table 1-16 verboseToZip Arguments

Argument	Definition
inputFile	Required. Legacy Java class-load log containing `[Loaded ...]` records. JDK 21 `java -verbose:class` output is not accepted by the current utility.
zipFileToCreate	Required. Name of the `.zip` file to be created. The resulting `.zip` file is created in the directory in which you run the program.

WebLogicMBeanMaker

The `WebLogicMBeanMaker` utility takes an XML MBean Description File (MDF) and outputs some intermediate Java files, including an MBean interface, an MBean implementation, and an associated MBean information file. Together, these intermediate files form the MBean type for a custom security provider.
See Understand What the WebLogic MBeanMaker Provides in *Developing Security Providers for Oracle WebLogic Server*.

Syntax

```
$ java -DMDF=xmlfile -Dfiles=filesdir -DcreateStubs=true|false
weblogic.management.commo.WebLogicMBeanMaker
```

Table 1-17 describes the arguments that are passed to `WebLogicMBeanMaker`.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 1-17 WebLogicMBeanMaker Arguments

Argument	Description
`-DMDF=xmlfile`	Specifies that the WebLogic MBeanMaker utility should translate the MDF file, represented by `xmlfile`, into code. Whenever `xmlfile` is specified, a new set of output files is generated.
`-Dfiles=filesdir`	Specifies the location, represented by `filesdir`, where the WebLogic MBeanMaker utility places the intermediate files for the MBean type.
`-DcreateStubs=true|false`	Specifies whether the WebLogic MBeanMaker utility overwrites any existing MBean implementation file.

wlappc

The `wlappc` utility compiles and validates a Jakarta EE EAR file, an EJB JAR file, or a WAR file for deployment.

See Building Modules and Applications Using wlappc in *Developing Applications for Oracle WebLogic Server*.

wlcompile

Use the `wlcompile` Ant task to invoke the `javac` compiler to compile your application's Java files in a split development directory structure.

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 for Oracle WebLogic Server*.

wldeploy

The `wldeploy` Ant task enables you to perform `weblogic.Deployer` tool functions using attributes specified in an Ant task.
See Deployer. For more information about `wldeploy`, see Deploying and Packaging from a Split Development Directory in *Developing Applications for Oracle WebLogic Server*.

wlpackage

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

wlserver

The `wlserver` Ant task enables you to start, reboot, shutdown, or connect to a WebLogic Server instance.

Copyright © 2007, 2026, Oracle and/or its affiliates.

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 for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

2

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 Node Manager. For production systems, use the generated domain startup scripts or Node Manager. These paths initialize the installation and domain environment, including the launcher and patch settings. Direct invocation of `java weblogic.Server` requires you to supply the corresponding environment explicitly.

The following sections explain how to use the `weblogic.Server` class to start WebLogic Server:

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

For information about using the Node Manager to start an instance of WebLogic Server, see Using Node Manager to Control Servers in the *Administering Node Manager for Oracle WebLogic Server*.

Required Environment and Syntax for weblogic.Server

Before you can use the `weblogic.Server` class to start a WebLogic Server instance, you must install WebLogic Server, set the `CLASSPATH` environment variable, and include a Java Virtual Machine (JVM) in your `PATH` environment variable.

Environment

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

1. Install and configure the WebLogic Server software, as described in *Installing and Configuring Oracle WebLogic Server and Coherence*.
2. For normal server startup, use the generated domain startup scripts or Node Manager. Before invoking `weblogic.Server` directly, initialize the command shell by running `WL_HOME\server\bin\setWLSEnv.cmd` on Windows or sourcing `WL_HOME/server/bin/setWLSEnv.sh` on UNIX. Alternatively, explicitly provide the required WebLogic classes on the Java class path. See Modifying the Classpath.
3. Include a Java Virtual Machine (JVM) in your `PATH` environment variable. You can use any JVM that is listed in Supported Configurations.

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

Modifying the Classpath

Installing WebLogic Server does not initialize the class path of an existing command shell. The `setWLSEnv` script initializes the WebLogic environment for that shell.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Apply product patches using the instructions supplied with the patch and the supported patching procedure for this release. See Patching Your Environment Using OPatch in *Patching with OPatch*.

> **Note**
>
> `setDomainEnv` is designed to be sourced from other scripts, such as the `startWebLogic` script. `setDomainEnv` should not be called directly from within an interactive shell. Doing so can cause unpredictable issues in the domain.

If you use Derby, the open-source all-Java database management system included with Oracle WebLogic Server for use by the sample applications and code examples, include the following files on the classpath:

- *WL_HOME*`/common/derby/lib/derbyclient.jar` - for the driver on the client side
- *WL_HOME*`/common/derby/lib/derbynet.jar` and *WL_HOME*/common/derby/lib/derby.jar - for running the Derby network server

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

WL_HOME`/server/lib/wlepool.jar`

WL_HOME`/server/lib/wleorb.jar`

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

Understand the default sequence of operations that occur when a WebLogic Server instance is started without any options having been passed to the `weblogic.Server` class.

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

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

Copyright © 2007, 2026, Oracle and/or its affiliates.

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

b. If there are multiple server instances defined in `config/config.xml`:

- If an Administration Server is defined, it looks for the server with that name.
- If an Administration Server is not defined, it looks for a server configuration named `myserver`. If it finds such a server configuration, it starts the `myserver` instance.
- If it does not find a server named `myserver`, WebLogic Server exits the `weblogic.Server` process and generates an error message.

3. If there is no `config.xml` file in the current directory, WebLogic Server prompts you to create one. If you respond `y`, WebLogic Server does the following:

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

Any options that you specify are persisted to the `config.xml` file. For example, if you specify `-Dweblogic.ListenPort=8001`, then WebLogic Server saves `8001` in the `config.xml` file. For any options that you do not specify, the server instance uses default values.

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

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

WebLogic Server also encrypts and stores your username and password in a *server_name*`/security/boot.properties` file, which enables you to bypass the login prompt during subsequent instantiations of the server. See Boot Identity Files in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

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

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

weblogic.Server Configuration Options

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

- JVM Parameters
- Location of Configuration Data
- Options that Override a Server's Configuration

Copyright © 2007, 2026, Oracle and/or its affiliates.

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

Unless you are creating a new domain as described in Using the weblogic.Server Command Line to Create a Domain, all startup options apply to the current server instantiation; they do not modify the persisted values in an existing `config.xml` file. Use the WebLogic Remote Console or WebLogic Scripting Tool (WLST) to modify the `config.xml` file. See Creating Domains Using WLST Offline in *Understanding the 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.

JVM Parameters

Table 2-1 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 Supported Configurations.

Table 2-1 Frequently Used Options for Setting JVM Parameters

Option	Description
`-Xms and -Xmx`	Specify the minimum and maximum values (in megabytes) for Java heap memory. For example, to request a 256 MB initial heap and a 512 MB maximum heap, start the server with `-Xms256m -Xmx512m`. These are example values, not the defaults generated for a domain; the generated startup scripts for this release normally initialize both values to 512 MB, subject to template and user overrides. The values assigned to these parameters can dramatically affect the performance of your WebLogic Server and are provided here only as examples. In a production environment you should carefully consider the correct memory heap size to use for your applications and environment.
`-classpath`	The minimum content for this option is described under Modifying the Classpath. Instead of using this argument, you can use the `CLASSPATH` environment variable to specify the classpath.
`-client -server`	Used by some JVMs to start a HotSpot virtual machine, which enhances performance. For a list of JVMs that can be used with WebLogic Server, see Supported Configurations.
`-Dfile.encoding=`*`charset`*	The supported JDKs use UTF-8 as the default charset. Use `-Dfile.encoding=UTF-8` to set that value explicitly. Use `-Dfile.encoding=COMPAT` only when compatibility with the operating-system-dependent default is required. See `Charset.defaultCharset` and Supported Encodings in the documentation for the JDK used to run the server. For example: `java -Dfile.encoding=UTF-8 weblogic.Server`

Copyright © 2007, 2026, Oracle and/or its affiliates.

Location of Configuration Data

All server instances must have access to configuration data. Table 2-2 provides options for indicating the location of this data.

Table 2-2 Options for Indicating the Location of Configuration Data

Option	Description
`-Dweblogic.home=`*`WL_HOME`*	Specifies the location of the WebLogic home directory, which contains essential information. By default, `weblogic.Server` determines the location of the WebLogic home directory based on values in the classpath.
`-Dweblogic.RootDirectory=`*`path`*	Specifies the server's root directory. See A Server's Root Directory in *Understanding Domain Configuration for Oracle WebLogic Server*. By default, the root directory is the directory from which you issue the start command.
`-Dweblogic.management.GenerateDefaultConfig=true`	Prevents the `weblogic.Server` class from prompting for confirmation when creating a `config.xml` file. Valid only if you invoke `weblogic.Server` in an empty directory. See Default Behavior.
`-Dweblogic.Domain=`*`domain`*	Specifies the name of the domain. If you are using `weblogic.Server` to create a domain, you can use this option to give the domain a specific name. In addition, this option supports a directory structure that WebLogic Server required in releases prior to 7.0 and continues to support in current releases. Prior to 7.0, all configuration files were required to be located in the following pathname: `.../config/`*`domain_name`*`/config.xml` In this pathname, *`domain_name`* is the name of the domain. If your domain's configuration file conforms to that pathname, and if you invoke the `weblogic.Server` command from a directory other than `config/`*`domain_name`*, you can include the `-Dweblogic.Domain=`*`domain`* argument to cause WebLogic Server to search for a `config.xml` file in a pathname that matches `config/`*`domain_name`*`/config.xml`.

For information on how a Managed Server retrieves its configuration data, see the `-Dweblogic.management.server` entry in Table 2-3.

The WebLogic Remote 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.

Example

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

Copyright © 2007, 2026, Oracle and/or its affiliates.

with SimpleServer. The command itself is issued from the `D:\` directory after running *`WL_HOME`*`\server\bin\setWLSEnv.cmd`:

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

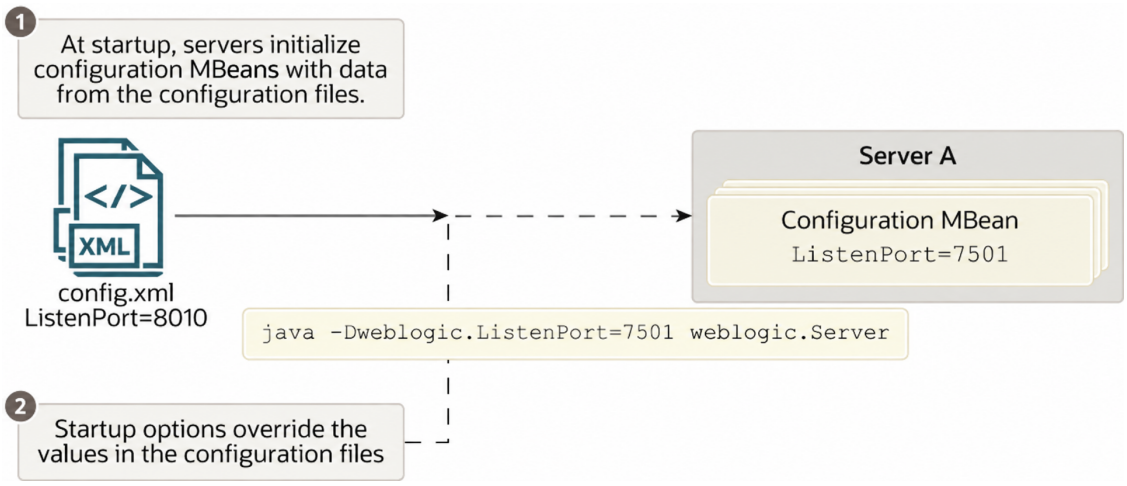
Options that Override a Server's Configuration

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

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

For example, in a production environment, your organization might have a policy against modifying the domain's `config.xml` file, but you need to shut down the Administration Server and restart it using a temporary listen port. In this case, when you use the `weblogic.Server` command to start the Administration Server, you can include the `-Dweblogic.ListenPort=7501` startup option to change the listen port for the current server session. The server instance initializes its configuration MBeans from the `config.xml` file but substitutes `7501` as the value of its listen port. When you subsequently restart the server without passing the startup option, it will revert to using the value from the `config.xml` file, `8010`. (See Figure 2-1.)

Figure 2-1 Overriding config.xml Values



The following options temporarily override a server's configuration:

Server Communication

Table 2-3 describes the options for configuring how servers communicate.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-3 Options for Configuring Server Communication

Option	Description
`-Dweblogic.management.server=[`*`protocol`*`://]`*`Admin-host:port`*	Starts a server instance as a Managed Server and specifies the Administration Server that will configure and manage the server instance. The domain's configuration file does not specify whether a server configuration is an Administration Server or a Managed Server. You determine whether a server instance is in the role of Administration Server or Managed Server with the options that you use to start the instance. If you omit the `-Dweblogic.management.server` option in the start command, the server starts as an Administration Server (although within a given domain, there can be only one active Administration Server instance). Once an Administration Server is running, you must start all other server configurations as Managed Servers by including the `-Dweblogic.management.server` option in the start command. For *`protocol`*, specify `HTTP`, `HTTPS`, `T3`, or `T3S`. The `T3S` and `HTTPS` protocols require you to enable SSL on the Managed Server and the Administration Server and specify the Administration Server's SSL listen port. **Note:** Regardless of which protocol you specify, the initial download of a Managed Server's configuration is over HTTP or HTTPS. After the RMI subsystem initializes, the server instance can use the T3 or T3S protocol. For `Admin-host`, specify localhost or the DNS name or IP address of the machine where the Administration Server is running. For `port`, specify the Administration Server's listen port. If you set up the domain-wide administration port, `port` must specify the domain-wide administration port. See Configuring Managed Server Connections to the Administration Server in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.
`-Dweblogic.ListenAddress=`*`host`*	Specifies the address at which this server instance listens for requests. The *`host`* value must be either the DNS name or the IP address of the computer that is hosting the server instance. This startup option overrides any listen address value specified in the `config.xml` file. The override applies to the current server instantiation; it does not modify the value in the `config.xml` file. Use any of the administration tools listed in Summary of System Administration Tools and APIs in *Understanding Oracle WebLogic Server* to modify the `config.xml` file. See Creating Domains Using WLST Offline in *Understanding the WebLogic Scripting Tool*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-3 (Cont.) Options for Configuring Server Communication

Option	Description
`-Dweblogic.ListenPort=`*`portnumber`*	Enables and specifies the plain-text (non-SSL) listen port for the server instance. This startup option overrides any listen port value specified in the `config.xml` file. The override applies to the current server instantiation; it does not modify the value in the `config.xml` file. Use any of the administration tools listed in Summary of System Administration Tools and APIs in *Understanding Oracle WebLogic Server* to modify the `config.xml` file. The default listen port is 7001. See Creating Domains Using WLST Offline in *Understanding the WebLogic Scripting Tool*.
`-Dweblogic.ssl.ListenPort=`*`portnumber`*	Enables and specifies the port at which this WebLogic Server instance listens for SSL connection requests. This startup option overrides any SSL listen port value specified in the `config.xml` file. The override applies to the current server instantiation; it does not modify the value in the `config.xml` file. Use any of the administration tools listed in Summary of System Administration Tools and APIs in *Understanding Oracle WebLogic Server* to modify the `config.xml` file. The default SSL listen port is 7002. See Creating Domains Using WLST Offline in *Understanding the WebLogic Scripting Tool*.
`-Dweblogic.management.discover={true|false}`	**Note:** This option was removed as of WebLogic Server 9.0. Determines whether an Administration Server recovers control of a domain after the server fails and is restarted. A `true` value causes an Administration Server to communicate with all known Managed Servers and inform them that the Administration Server is running. A false value prevents an Administration Server from communicating with any Managed Servers that are currently active in the domain. **Tip:** Specify `false` for this option only in the development environment of a single server. Specifying `false` can cause server instances in the domain to have an inconsistent set of deployed modules. In WebLogic Server 9.0, this command is deprecated because if an Administration Server stops running while the Managed Servers in the domain continue to run, each Managed Server will periodically attempt to reconnect to the Administration Server at the interval specified by the `ServerMBean` attribute `AdminReconnectIntervalSecs`. See *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

The WebLogic Remote 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.

SSL

Each WebLogic Server instance uses an instance of `weblogic.management.configuration.SSLMBean` to represent its SSL configuration.

Copyright © 2007, 2026, Oracle and/or its affiliates.

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.

Table 2-4 describes the options for configuring a server to communicate using Secure Sockets Layer (SSL).

> **Note**
>
> As of WebLogic Server version 12.1.1, JSSE is the only SSL implementation that is supported. The Certicom-based SSL implementation is removed and is no longer supported in WebLogic Server.

Table 2-4 Options for Configuring SSL

Option	Description
`-Dweblogic.security.SSL.ignoreHostnameVerification=true`	Disables host name verification, which enables you to use the demonstration digital certificates that are shipped with WebLogic Server. By default, when a WebLogic Server instance is in the role of SSL client (it is trying to connect to some other server or application via SSL), it verifies that the host name that the SSL server returns in its digital certificate matches the host name of the URL used to connect to the SSL server. If the host names do not match, the connection is dropped. If you disable host name verification, either by using this option or by modifying the server's configuration in the `config.xml` file, the server instance does not verify host names when it is in the role of SSL client. **Note:** Oracle does not recommend using the demonstration digital certificates or turning off host name verification in a production environment. This startup option overrides any host name verification setting in the `config.xml` file. The override applies to the current server instantiation; it does not modify the value in the `config.xml` file. Use the WebLogic Remote Console or WLST to modify the `config.xml` file. See Using Host Name Verification in *Administering Security for Oracle WebLogic Server*.
`-Dweblogic.security.SSL.HostnameVerifier=`*`hostnameverifierimplementation`*	Specifies the name of a custom host name verifier class. The class must implement the `weblogic.security.SSL.HostnameVerifier` interface.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-4 (Cont.) Options for Configuring SSL

Option	Description
`-Dweblogic.security.SSL.sessionCache.ttl=`*`sessionCacheTimeToLive`*	Modifies the default server-session time-to-live for SSL session caching. The *`sessionCacheTimeToLive`* value specifies (in milliseconds) the time to live for the SSL session. The default value is 90000 milliseconds (90 seconds). This means if a client accesses the server again (via the same session ID) within 90 seconds, WebLogic Server will use the existing SSL session. You can change this value by setting `-Dweblogic.security.SSL.sessionCache.ttl` in the server startup script. For *`sessionCache.ttl`*: • The minimum value is `1` • The maximum value is `Integer.MAX_VALUE` • The default value is `90000`
`-Dweblogic.security.SSL.CertificateCallback=`*`callback-handler`*	Specifies a certificate callback handler class, which evaluates details contained in the end-user certificate passed in a secure connection request to WebLogic Server. Depending on the details contained in the certificate, the callback handler returns a `true` or `false`, which determines whether authentication is successful. **Note:** If you use a certificate callback implementation in WebLogic Server, a callback is generated whenever a request is received over a secure port. As a result, using certificate callbacks may impose a performance overhead that should be taken into consideration. See Checking the Validity of End User Certificates in *Administering Security for Oracle WebLogic Server*.
`-Dweblogic.management.pkpassword=`*`pkpassword`*	Specifies the password for retrieving SSL private keys from an encrypted flat file. Use this option if you store private keys in an encrypted flat file.
`-Dweblogic.security.SSL.trustedCAKeyStore=`*`path`*	Specifies the trusted keystore to use in either of these use cases: • Starting a Managed Server that downloads the initial configuration from the Administration Server over SSL • Running a WebLogic client, such as WLST, that connects to a WebLogic Server server instance over SSL The *`path`* value must be the full path to the JKS or PKCS12 trust keystore file. Note that if the keystore is type PKCS12, then a password is required. You can specify the password using the `-Dweblogic.security.SSL.trustedCAKeyStorePassPhrase` as described below. Oracle recommends that you do not use the demonstration certificate authorities in any type of production deployment. See Configuring SSL in *Administering Security for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-4 (Cont.) Options for Configuring SSL

Option	Description
`-Dweblogic.security.SSL.trustedCAKeyStorePassPhrase=`*`password`*	Specifies the password to use if you specified a PKCS12 trust keystore using the `-Dweblogic.security.SSL.trustedCAKeyStore` property. The password is not necessary for JKS trust keystores. When running a WebLogic client such as WLST, if the type of trust keystore specified by `-Dweblogic.security.SSL.trustedCAKeyStore` is not JKS (for example, it is PKCS12) and you do not specify the `-Dweblogic.security.SSL.trustedCAKeyStorePassPhrase` property, then WebLogic prompts you to enter the password.
`-Dweblogic.security.SSL.protocolVersion=`*`protocol`*	Specifies the protocol that is used for SSL connections. The *`protocol`* value can be one of the following: • SSL3 — Only SSL v3.0 messages are sent and accepted. • TLS1 — Enables any protocol starting with TLS for messages that are sent and accepted; for example, TLS v1.0, TLS v1.1, TLS v1.2, and TLS v1.3. **Note** – Support for TLS v1.0 and v1.1 is deprecated. Oracle strongly recommends that you do not use TLS v1.0 and v1.1. In addition, these versions may be disabled by default in certain JDK updates by the underlying JSSE provider. • ALL — Depends on the JSSE provider and JDK version. If this legacy property is not set, WebLogic Server enables the protocols supported by the configured JSSE provider at or above the default minimum protocol version, `TLSv1.2`. Use `weblogic.security.SSL.minimumProtocolVersion` to select a different minimum. See Using the weblogic.security.SSL.protocolVersion System Property in *Administering Security for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-4 (Cont.) Options for Configuring SSL

Option	Description
`-Dweblogic.security.SSL.minimumProtocolVersion=`*`protocol`*	Specifies the *minimum protocol version* that is used for SSL connections. The *`protocol`* value can be one of the following: • SSLv3 — Specifies SSL v3.0 as the minimum protocol version enabled in SSL connections. • TLSv1 — Specifies TLS v1.0 as the minimum protocol version enabled in SSL connections. • TLSv*x.y* — Specifies TLS v*x.y* as the minimum protocol version enabled in SSL connections, where *x* is between 1 and 9, and *y* is between 0 and 9, inclusive. **Note** • TLS v1.2 is the default minimum protocol version configured in WebLogic Server. Oracle recommends the use of TLS v1.2 or later in a production environment. WebLogic Server logs a warning if the TLS version is set below 1.2. • Support for TLS v1.0 and v1.1 is deprecated. Oracle strongly recommends that you do not use TLS v1.0 and v1.1. In addition, these versions may be disabled by default in certain JDK updates by the underlying JSSE provider. • TLSv1.3 is available when it is supported by the configured JDK/JSSE provider. Use a JDK listed in Supported Configurations for this WebLogic Server release. See Using the weblogic.security.SSL.minimumProtocolVersion System Property in *Administering Security for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-4 (Cont.) Options for Configuring SSL

Option	Description
`-Dweblogic.security.ssl.sslcontext.protocol=`*`protocol`*	Specifies the `javax.net.ssl.SSLContext` algorithm for the JSSE provider. For some JSSE providers, there is a correlation between the `javax.net.ssl.SSLContext` algorithm and the initially enabled SSL/TLS protocols. If the JSSE provider configured in your system interprets the protocol parameter differently, you may need to set this property. Refer to the vendor-specific documentation for the correlations between the `javax.net.ssl.SSLContext` setting and the enabled SSL/TLS protocols. **Note** When using the IBM JSSE provider, WebLogic Server attempts to select a `javax.net.ssl.SSLContext` algorithm equivalent to the default TLS. Standard supported values are `SSL`, `SSLv3`, `TLS` (the default), `TLSv1`, `TLSv1.1`, `TLSv1.2`, and `TLSv1.3`. WebLogic Server does not support `SSLv2`. `TLSv1.3` is available when it is supported by the configured JDK/JSSE provider. Use a JDK listed in Supported Configurations for this WebLogic Server release. See Using the weblogic.security.ssl.sslcontext.protocol System Property in *Administering Security for Oracle WebLogic Server*.
`-Dweblogic.security.TrustKeyStore=`*`Keystore`*	Specifies which keystore the server should trust while starting. Specify one of: • `DemoTrust` if using the demo identity and trust certificates. • `DomainTrust` if using the Domain Keystore certificates. When using Domain Keystores, you must specify `weblogic.security.TrustKeyStore` even if you're not using an HTTPS or T3S URL to communicate with the Administration Server. • `JavaStandardTrust` if using the Java default truststore. • `CustomTrust` if using a custom trust keystore. The server will use the keystore you specified with the `-Dweblogic.security.CustomTrustKeyStoreType`, `-Dweblogic.security.CustomTrustKeyStoreFileName`, and `-Dweblogic.security.CustomTrustKeyStorePassPhrase` properties. See Configuring Keystores in *Administering Security for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-4 (Cont.) Options for Configuring SSL

Option	Description
`-Dweblogic.security.CustomTrustKeyStoreFileName`	Specifies the name and location of the custom keystore. It must match exactly the value as specified in the `ServerMBean.CustomTrustKeyStoreFileName` attribute in the domain's `config.xml` file. **Note**: If using this property, you must also specify `-Dweblogic.security.CustomTrustKeyStoreType`.
`-Dweblogic.security.CustomTrustKeyStoreType`	Specifies the type of keystore.
`-Dweblogic.security.CustomTrustKeyStorePassPhrase`	Specifies the password that was used to secure the custom keystore. Only use this option if the custom keystore is protected by a password.

WebLogic Remote 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.

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`, described in *MBean Reference for Oracle WebLogic Server*. 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 WebLogic Remote 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.

HTTP Strict Transport Security

HTTP Strict Transport Security (HSTS) is a web security policy mechanism that allows a web server to be configured so that web browsers, or other user agents, can access the server using only secure connections, such as HTTPS. Web servers declare this policy using the Strict-Transport-Security HTTP response header field.

Copyright © 2007, 2026, Oracle and/or its affiliates.

For more information about HSTS, see Using HTTP Strict Transport Security in *Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.

Table 2-5 describes the system properties that you add to WebLogic Server domain start up scripts to enable HSTS and customize the response header.

Table 2-5 Options for Configuring HSTS

Option	Description
`-Dweblogic.http.headers.enableHSTS={true|false}`	Specifies whether the server is configured to use HSTS. The default value of this system property is `false`, indicating that WebLogic Server will NOT send the HSTS header by default. Set this property to `true` to enable HSTS. When enabled, the following default HSTS header is sent with all responses to secure (HTTPS) requests: `Strict-Transport-Security: max-age=31536000; includeSubDomains; preload` The values specified in this response are the minimum values required by the HSTS preload submission web site https://hstspreload.org/. You can customize these values using the system properties described in the following rows of this table.
`-Dweblogic.http.headers.hsts.maxage=`*`max-age-seconds`*	Specifies the time, in seconds, that the browser remembers that a site is only to be accessed using HTTPS. The default value is 31536000 seconds (one year).
`-Dweblogic.http.headers.hsts.includesubdomains={true|false}`	Specifies whether the HSTS policy applies to this HSTS host as well as any subdomains of the host's domain name. If this directive is not specified, the property defaults to `true` and is included in the header response.
`-Dweblogic.http.headers.hsts.preload={true|false}`	Specifies that the site is requesting inclusion in the HTTP Strict Transport Security (HSTS) preload list, which is a list of sites that are hardcoded into Chrome (and other browsers) as using HTTPS only. If this directive is not specified, the property defaults to `true` and is included in the header response. Ensure that your site meets all the necessary requirements before including the `preload` directive in your response header. To ensure your site is successfully preloaded, request submission at `https://hstspreload.org/`.

Security

Table 2-6 describes the options for configuring general security parameters.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-6 Options for General Security Parameters

Option	Description
`-Dweblogic.management.username=`*`username`*	Specifies the username under which the server instance will run. As of WebLogic Server 12.1.1, the boot username property `weblogic.management.username` has been deprecated and will be removed in a future release, and you will no longer be able to specify the username in the command for starting WebLogic Server in production mode. As an alternative, Oracle recommends that you use the `boot.properties` file to specify the boot username for WebLogic Server. See Boot Identity Files in *Administering Server Startup and Shutdown for Oracle WebLogic Server*. See Provide User Credentials to Start and Stop Servers in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.
`-Dweblogic.management.password=`*`password`*	Specifies the user password. As of WebLogic Server 12.1.1, the boot password system property `weblogic.management.password` has been deprecated and will be removed in a future release, and you will no longer be able to specify the password in the command for starting WebLogic Server in production mode. As an alternative, Oracle recommends that you use the `boot.properties` file to specify the boot password for WebLogic Server. See Boot Identity Files in *Administering Server Startup and Shutdown for Oracle WebLogic Server*. See Provide User Credentials to Start and Stop Servers in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-6 (Cont.) Options for General Security Parameters

Option	Description
`-Dweblogic.system.StoreBootIdentity=true`	Creates a `boot.properties` file in `DOMAIN_HOME/servers/server-name/security`. The file contains the username and an encrypted version of the password that was used to start the server. Do not specify this argument in a server's `ServerStartMBean`. See Specifying User Credentials When Starting a Server with the Node Manager in *Administering Server Startup and Shutdown for Oracle WebLogic Server*. Oracle recommends that you do not add this argument to a startup script. Instead, use it only when you want to create a `boot.properties` file. See Boot Identity Files in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.
`-Dweblogic.system.BootIdentityFile=`*`filename`*	Specifies a boot identity file that contains a username and password. The *`filename`* value must be the fully qualified pathname of a valid boot identity file. For example: `-Dweblogic.system.BootIdentityFile=C:\Oracle\Middleware\Oracle_Home\user_projects\domains\mydomain\servers\myserver\security\boot.properties` If you do not specify this system property, server startup looks for `boot.properties` in `DOMAIN_HOME/servers/server-name/security`. If no boot identity file exists when the server starts, the server prompts for a username and password.
`-Dweblogic.system.RemoveBootIdentity=true`	Removes the boot identity file after a server starts.
`-Dweblogic.security.anonymousUserName=`*`name`*	Assigns a user ID to anonymous users. By default, all anonymous users are identified with the string `<anonymous>`. To emulate the security behavior of WebLogic Server 6.x, specify `guest` for the *`name`* value and create a user named `guest` in your security realm. See Users, Groups, and Security Roles in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-6 (Cont.) Options for General Security Parameters

Option	Description
`-Djava.security.manager` `-Djava.security.policy[=]=`*`filename`*	On JDK 21, these Java SE options enable the Java Security Manager and specify a relative or fully qualified path to a Java security policy file. On JDK 25, the Java Security Manager is permanently disabled: `-Djava.security.manager` causes the JVM to report an error and exit, and `java.security.policy` is unsupported. To use the WebLogic Server sample policy file, specify *`WL_HOME`*`\server\lib\weblogic.policy`. On JDK 21, using `-Djava.security.policy==`*`filename`* (note the double equal sign (==)) causes the policy file to override any default security policy. When JACC is enabled, this property causes WebLogic Server to ignore any policy files that are used for servlet and EJB authorization. A single equal sign (=) causes the policy file to be appended to an existing security policy. **Note:** The WebLogic JACC provider does not require the Java Security Manager to be enabled. See Using the Java Security Manager to Protect WebLogic Resources in *Developing Applications with the WebLogic Security Service*.
`-Dweblogic.security.fullyDelegateAuthorization=true`	By default, roles and security policies cannot be set for an EJB or Web application through the WebLogic Remote Console unless security constraints were defined in the deployment descriptor for the EJB or Web application. Use this option when starting WebLogic Server to override this problem. This startup option does not work with EJBs or EJB methods that use `<unchecked>` or `<restricted>` tags or Web applications that do not have a role-name specified in the `<auth-constraint>` tag.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-6 (Cont.) Options for General Security Parameters

Option	Description
`-Dweblogic.management.anonymousAdminLookupEnabled=true`	Enables you to retrieve an `MBeanHome` interface without specifying user credentials. The `MBeanHome` interface is part of the WebLogic Server JMX API. If you retrieve `MBeanHome` without specifying user credentials, the interface gives you read-only access to the value of any MBean attribute that is not explicitly marked as protected by the Weblogic Server MBean authorization process. This startup option overrides the Anonymous Admin Lookup Enabled setting on the *domain_name* > Security > General page in the WebLogic Remote Console. By default, the `MBeanHome` API allows access to MBeans only for WebLogic users who are in one of the default security roles. See Users, Groups, an Security Roles in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*.
`-Dweblogic.security.identityAssertionTTL=`*seconds*	Configures the number of seconds that the Identity Assertion cache stores a Subject. When using an Identity Assertion provider (either for an X.509 certificate or some other type of token), Subjects are cached within the server. This greatly enhances performance for servlets and EJB methods with `<run-as>` tags as well as for other places where identity assertion is used but not cached (for example, signing and encrypting XML documents). There might be some cases where this caching violates the desired semantics. The default Identity Assertion cache time-to-live is 300 seconds. Setting this property to `-1` disables the cache. Setting a high value generally improves the performance of identity assertion, but makes the Identity Assertion provider less responsive to changes in the configured Authentication provider. For example, a change in the user's group will not be reflected until the Subject is flushed from the cache and recreated.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-6 (Cont.) Options for General Security Parameters

Option	Description

`-Djakarta.security.jacc.PolicyConfigurationFactory.provider=weblogic.security.jacc.simpleprovider.PolicyConfigurationFactoryImpl`
`-Djakarta.security.jacc.policy.provider=weblogic.security.jacc.simpleprovider.SimpleJACCPolicy`
`-Dweblogic.security.jacc.RoleMapperFactory.provider=weblogic.security.jacc.simpleprovider.RoleMapperFactoryImpl`
	These properties select the built-in WebLogic JACC providers. When the first two properties select the built-in providers shown here, the third property is optional. For compatibility, WebLogic Server also accepts the legacy `javax.security.jacc.PolicyConfigurationFactory.provider` and `javax.security.jacc.policy.provider` aliases. Do not specify conflicting values under the `jakarta` and `javax` names. When these providers are in use, JACC handles authorization decisions for the EJB and servlet containers for external applications. Any other authorization decisions for internal applications are handled by the authorization in the WebLogic Security framework.
	Note: JACC authorization does not require the use of Java SE security.
	When starting, WebLogic Server attempts to locate and instantiate the classes specified by the JACC startup properties and fails if it cannot find or instantiate them (if, for example, the files specified by the startup properties are not valid classes).
	See Using the Java Authorization Contract for Containers in *Developing Applications with the WebLogic Security Service*

`-Dweblogic.security.ldap.maxSize=`*`max-bytes`*	Limits the size of the data file used by the embedded LDAP server. When the data file exceeds the specified size, WebLogic Server eliminates from the data file space occupied by deleted entries.
`-Dweblogic.security.ldap.changeLogThreshold=<`*`number of entries`*`>`	Limits the size of the change log file used by the embedded LDAP server. When the change log file exceeds the specified number of entries, WebLogic Server truncates the change log by removing all entries that have been sent to all Managed Servers.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-6 (Cont.) Options for General Security Parameters

Option	Description
`-Dweblogic.security.providers.authentication.ldap.socketTimeout=`*seconds*	Sets a timeout value for the LDAP Authentication provider connection to an LDAP server. If multiple LDAP servers are specified in the `LDAPServerMBean.Host` attribute, the socket timeout applies to each individual LDAP server connection attempt that is made. The default value is 0, which sets no socket timeout on connections. Note that the `LDAPServerMBean.ConnectTimeout` attribute sets the timeout limit for *all* connection attempts that are made. Typically the socket timeout is used in conjunction with the connect timeout and the parallel connect delay. See Configuring Failover for LDAP Authentication Providers in *Administering Security for Oracle WebLogic Server*.

The Edit Tree perspective of the WebLogic Remote Console does not show values set on the command line because it reflects `config.xml`. Use the Configuration View Tree perspective to see the server's current effective values. See Verifying Attribute Values That Are Set on the Command Line.

Message Output and Logging

Table 2-7 describes options for configuring a server instance's message output.

Table 2-7 Options for Configuring Message Output

Option	Description
`-Dweblogic.Stdout="`*filename*`"`	Redirects the server and JVM standard output stream to a file. You can specify a pathname that is fully qualified or relative to the WebLogic Server root directory.
`-Dweblogic.Stderr="`*filename*`"`	Redirects the server and JVM standard error stream to a file. You can specify a pathname that is fully qualified or relative to the WebLogic Server root directory. See Redirecting JVM Output in *Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server*.
`-Dweblogic.AdministrationMBeanAuditingEnabled={true|false}`	Determines whether the Administration Server emits configuration auditing log messages when a user changes the configuration or invokes management operations on any resource within a domain. By default, the Administration Server does not emit configuration auditing messages. See Enable Configuration Auditing in the *Oracle WebLogic Remote Console Online Help*.

The Edit Tree perspective of the WebLogic Remote Console does not show values set on the command line because it reflects `config.xml`. Use the Configuration View Tree perspective to see the server's current effective values. See Verifying Attribute Values That Are Set on the Command Line.

Copyright © 2007, 2026, Oracle and/or its affiliates.

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, see `LogMBean` in *MBean Reference for Oracle WebLogic Server*. 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 Edit Tree perspective of the WebLogic Remote Console does not show values set on the command line because it reflects `config.xml`. Use the Configuration View Tree perspective to see the server's current effective values. See Verifying Attribute Values That Are Set on the Command Line.

Clusters

Table 2-8 describes options for configuring additional attributes of a cluster.

Table 2-8 Options for Configuring Cluster Attributes

Option	Description
`-Dweblogic.cluster.multicastAddress`	Applies only to a cluster whose `ClusterMessagingMode` is `multicast`. Multicast messaging is provided for backward compatibility; the default cluster messaging mode is `unicast`. For a multicast cluster, this option overrides the `MulticastAddress` value in `config.xml` for the current server process. **Note:** The Edit Tree perspective of the WebLogic Remote Console does not show values set on the command line because it reflects `config.xml`. Use the Configuration View Tree perspective to see the server's current effective values. See Verifying Attribute Values That Are Set on the Command Line. All servers in the same multicast cluster must use the same multicast address.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Deployment

Table 2-9 describes options for configuring additional attributes for deployment.

Table 2-9 Options for Configuring Deployment Attributes

Option	Description
`-Dweblogic.deployment.IgnorePrepareStateFailures=true`	For a clustered server, overrides the default handling of a static deployment PREPARE failure and allows the server to continue to the `RUNNING` state. This option has no effect on a server that is not a cluster member. **Note:** This server level flag may cause inconsistent deployment behavior within clusters, such as issues with HttpSessionReplication or SFSB replication.

Other Server Configuration Options

Table 2-10 describes options for configuring additional attributes of a server instance.

Table 2-10 Options for Configuring Server Attributes

Option	Description
`-Dweblogic.Name=`*`servername`*	Specifies the name of the server instance that you want to start. The specified value must refer to the name of a server that has been defined in the domain's `config.xml` file.
`-Dweblogic.ProductionModeEnabled={true|false}`	Determines whether a server starts in production mode. For information on the default settings specified in production mode, see Understand How Domain Mode Affects the Default Security Configuration in *Securing a Production Environment for Oracle WebLogic Server*. To enable production mode, see Changing the Domain Mode in *Administering Security for Oracle WebLogic Server*. **Notes**: • As of WebLogic Server 14.1.2.0.0, enabling production mode automatically enables secured production mode as well. • It is recommended that you enable production mode using WLST Offline rather than `weblogic.ProductionModeEnabled`, whenever possible. You should only enable production mode from the command line on the Administration Server. • It is important to note that when `weblogic.ProductionModeEnabled` is set from the command line on the Administration Server, this value is propagated to all Managed Servers.
`-Dweblogic.securemode.SecureModeEnabled={true|false}`	Determines whether a server starts in secured production mode. **Note**: As of WebLogic Server 14.1.2.0.0, when production mode is enabled, secured production mode is implicitly enabled as well. You do not need to set both values. Unlike `weblogic.ProductionModeEnabled`, the `weblogic.securemode.SecureModeEnabled` option does not propagate automatically from the Administration Server to Managed Servers. You must set it separately on each server in the domain.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-10 (Cont.) Options for Configuring Server Attributes

Option	Description
`-Dweblogic.management.startupMode=MODE`	The argument *MODE* represents either of the following: • `STANDBY` — Starts a server and places it in the `STANDBY` state. See *Administering Server Startup and Shutdown for Oracle WebLogic Server*. To use this startup argument, the domain must be configured to use the domain-wide administration port. See Administration Port and Administrative Channel in *Administering Server Environments for Oracle WebLogic Server* and Configure the Domain-Wide Administration Port in the *Oracle WebLogic Remote Console Online Help*. • `ADMIN` — Starts a server and places it in the `ADMIN` state. See *Administering Server Startup and Shutdown for Oracle WebLogic Server*. Specifying this system property overrides the `StartupMode` value read from `config.xml` for the current server process. The system property takes precedence over the configured value and cannot be changed at runtime. It does not directly modify `config.xml`. However, starting and saving an edit session in the WebLogic Remote Console or WLST can persist the effective startup-mode value in `config.xml`.
`-Dweblogic.apache.xerces.maxentityrefs=numerical-value`	Limits the number of entities in an XML document that the WebLogic XML parser resolves. If you do not specify this option, the XML parser that WebLogic Server installs resolves 10,000 entity references in an XML document, regardless of how many an XML document contains.
`-Dweblogic.jsp.windows.caseSensitive=true`	Causes the JSP compiler on Windows systems to preserve case when it creates output files names. See Using the WebLogic JSP Compiler in *Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.
`-Dweblogic.servlet.optimisticSerialization=true`	When `optimistic-serialization` is turned on, WebLogic Server does not serialize-deserialize context and request attributes upon `getAttribute(name)` when the request is dispatched across servlet contexts. This means that you must make sure that the attributes common to Web applications are scoped to a common parent classloader (application scoped) or you must place them in the system classpath if the two Web applications do not belong to the same application. When `optimistic-serialization` is turned off (default value), WebLogic Server serialize-deserializes context and request attributes upon `getAttribute(name)` to avoid the possibility of ClassCastExceptions. The optimistic-serialization value can also be specified at domain level in the `WebAppContainerMBean`, which applies for all Web applications. The value in weblogic.xml, if specified, overrides the domain level value. The default value is false.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Table 2-10 (Cont.) Options for Configuring Server Attributes

Option	Description
`-Dweblogic.servlet.maxLoggingURILength=`*`length`*	By default, when using extended log format in HTTP access logs, the maximum logged URI length is 256 characters. If the URI exceeds that length, the logged URI is truncated. You can use this property to increase the length of the URI that is logged. See DNS Related Fields in *Administering Server Environments for Oracle WebLogic Server*.
`-Dweblogic.jdbc.qualifyRMName=false`	When set, restores pre-WebLogic Server 11*g*R1 (10.3.1) behavior of not qualifying the JTA registration name with the domain name.
`-Dweblogic.ScatteredReadsEnabled=true` and `-Dweblogic.GatheredWritesEnabled=true`	When each is set to true, increases efficiency during I/O in environments with high network throughput. These command options are used together to optimize WebLogic Server performance for use with Oracle Exalogic. See the *Oracle Exalogic Enterprise Deployment Guide*.
`-Dweblogic.replication.enableLazyDeserialization= true`	When set to true, increases efficiency with session replication. This command option is used to optimize WebLogic Server performance for use with Oracle Exalogic. See the *Oracle Exalogic Enterprise Deployment Guide*.
`-Dweblogic.resourcepool.max_test_wait_secs=`*`seconds`*	The amount of time, in seconds, WebLogic Server waits before considering a connection test failed. By default, a server instance is assigned a value of 10 seconds. If set to zero, the server instance waits indefinitely.
`-Dweblogic.wsee.client.ssl.usejdk=true`	When set to true, switches from `WlsSSLAdapter` to `JdkSSLAdapter`. By default, WebLogic Server Web services use the `weblogic.wsee.connection.transport.https.WlsSSLAdapter` class for the SSL adapter. Setting the flag to true forces the use of `JdkSSLAdapter` from `weblogic.wsee.connection.transport.https.JdkSSLAdapter`.
`-Dweblogic.http.URIDecodeEncoding=`*`charset-name`*	The argument *`charset-name`* specifies the encoding used by the WebLogic Server Web container to decode the URI of an HTTP request or to encode the `Location` header in an HTTP response. The default value is `UTF-8`. See Determining the Encoding of an HTTP Request in *Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.
`-Dweblogic.utils.http.requestparams.useArrayMap= true`	When set to true in the server startup command, HTTP request parameters are stored using an ArrayMap. By default, HTTP request parameters are stored in a TreeMap.

The WebLogic Remote Console does not display values that you set on the command line in all of its perspectives. For information on verifying the attribute values that you set, see Verifying Attribute Values That Are Set on the Command Line.

Copyright © 2007, 2026, Oracle and/or its affiliates.

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

The basic procedure to start a WebLogic Server instance is to run the `setWLSEnv` script to set the environment, change to the root directory of a domain, and enter the `java weblogic.Server` command.

Complete the following steps:

1. In a command shell, set up the required environment variables by running the following script:

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

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

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

2. In the command shell, change to the root of the domain directory, usually *ORACLE_HOME*`\user_projects\domains\`*DOMAIN_NAME*. For example, change to the *ORACLE_HOME*`\user_projects\domains\medrec` directory.

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

   ```
   java weblogic.Server
   ```

   > **Note**
   >
   > The password you use must be a string of at least 8 case-sensitive characters. The space character is not supported. See Creating a WebLogic Domain in *Creating WebLogic Domains Using the Configuration Wizard*.

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

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

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

   ```
   java -Dweblogic.Name=MedRecManagedServer
   -Dweblogic.management.server=localhost:7011
   weblogic.Server
   ```

> **Note**
>
> If you are using the demo certificates in a multi-server domain, Managed Server instances will fail to boot if you specify the fully-qualified DNS name of the Administration Server host machine, as in the argument *url-for-Administration Server*. See Limitation on CertGen Usage in *Administering Security for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Using the weblogic.Server Command Line to Create a Domain

You can invoke the `weblogic.Server` class to create a domain that contains a single server instance. However, you cannot invoke the `weblogic.Server` class either to add Managed Server instances to a domain, or to modify an existing domain.
As described in Default Behavior, 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.ListenPort=7001 weblogic.Server
   ```

After you enter this command, WebLogic Server asks whether you want to create a new configuration. If you enter `y`, WebLogic Server prompts for an administrative username and password and asks you to enter the password again for confirmation. It then creates and starts 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 with the username and password that you entered at the prompts.
- For the listen address of the Administration Server, you can use `localhost`, the IP address of the host computer, or the DNS name of the host computer.
- The Administration Server listens on port 7001.

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

- `config/config.xml`
- `DefaultAuthenticatorInit.ldift`, `DefaultRoleMapperInit.ldift`, and `SerializedSystemIni.dat`, which store basic security-related data.
- `boot.properties` file, which contains the username and password in an encrypted format. This file enables you to bypass the prompt for username and password when you start the server. See Boot Identity Files in *Administering Server Startup and Shutdown for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

- `startWebLogic.cmd` and `startWebLogic.sh`, that you can use to start subsequent instantiations of the server.

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

Verifying Attribute Values That Are Set on the Command Line

Attribute values that are set on the command line override but do not change the server's central configuration file. Therefore, some WebLogic Server administration tools will report attribute values that contradict what you set at the command line and the actual state of the server.

The Edit Tree perspective of the WebLogic Remote Console is generated from the `config.xml` configuration file and thus will not reflect `weblogic.Server` configuration options. You must use the Configuration View Tree perspective to see the current and effective state of the domain. See Administration Server Perspectives in *Oracle WebLogic Remote Console Online Help*.

You can also use WLST to see a server's local configuration MBean.

1. Complete the procedure described in Main Steps for Using WLST in Interactive or Script Mode in *Understanding the WebLogic Scripting Tool*.

   ```
   >java weblogic.WLST
   ```

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

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

   wls:/mydomain/serverConfig>
   ```

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

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

   '239.192.0.0'
   ```

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

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

   'Notice'
   ```

See *Understanding the WebLogic Scripting Tool* and Understanding WebLogic Server MBeans in *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*.

Copyright © 2007, 2026, Oracle and/or its affiliates.

Copyright © 2007, 2026, Oracle and/or its affiliates.