Oracle® Fusion Middleware

WebLogic Scripting Tool Command Reference

12*c* Release 1 (12.1.1)

E24490-01

December 2011

This document describes all of the commands that are available to use with the WebLogic Scripting Tool (WLST). This document includes WLST commands for WebLogic Server, as well as custom WLST commands that can be used to manage installed Oracle Fusion Middleware components.

ORACLE®

Oracle Fusion Middleware WebLogic Scripting Tool Command Reference, 12*c* Release 1 (12.1.1)

E24490-01

Contents

Preface

This preface describes the document accessibility features and conversions used in this guide—*WebLogic Scripting Tool Command Reference*.

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 customers have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

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 which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introduction and Roadmap

This section describes the audience for and contents and organization of this guide—*WebLogic Scripting Tool Command Reference*.

- Section 1.1, "Document Scope and Audience"
- Section 1.2, "Guide to This Document"
- Section 1.3, "Related Documentation"
- Section 1.4, "New and Changed WLST Features in This Release"

1.1 Document Scope and Audience

This document describes all of the commands that are available to use with the WebLogic Scripting Tool (WLST). This document includes WLST commands for WebLogic Server, as well as custom WLST commands that can be used to manage installed Oracle Fusion Middleware components.

> **Note:** Custom WLST commands for a given Oracle Fusion Middleware component are available for use only if the component is installed in the *ORACLE_HOME* directory.

This document is written for WebLogic Server administrators and operators who deploy Java EE applications using the Java Platform, Enterprise Edition (Java EE) from Oracle. It is assumed that readers are familiar with Web technologies and the operating system and platform where WebLogic Server is installed.

1.2 Guide to This Document

This document is organized as follows:

- This chapter, "Introduction and Roadmap," introduces the organization of this guide and lists related documentation.
- Chapter 2, "WebLogic Server WLST Online and Offline Command Reference," summarizes WebLogic Server WLST commands alphabetically and by online/offline usage.
- Chapter 3, "WLST Command and Variable Reference," provides detailed descriptions for each of the WebLogic Server WLST commands and variables.

1.3 Related Documentation

For information about how to use the WebLogic Scripting Tool, refer to *Oracle WebLogic Scripting Tool*.

WLST is one of several interfaces for managing and monitoring WebLogic Server. For information about the other management interfaces, see:

- "Using Ant Tasks to Configure and Use a WebLogic Server Domain" in *Developing Applications for Oracle WebLogic Server*, describes using WebLogic Ant tasks for starting and stopping WebLogic Server instances and configuring WebLogic domains.

- "Deployment Tools" in *Deploying Applications to Oracle WebLogic Server* describes several tools that WebLogic Server provides for deploying applications and stand-alone modules.

- *Administration Console Online Help* describes a Web-based graphical user interface for managing and monitoring WebLogic domains.

- *Creating WebLogic Domains Using the Configuration Wizard* describes using a graphical user interface to create a WebLogic domain or extend an existing one.

- *Creating Templates and Domains Using the Pack and Unpack Commands* describes commands that recreate existing WebLogic domains quickly and easily.

- *Developing Custom Management Utilities With JMX for Oracle WebLogic Server* describes using Java Management Extensions (JMX) APIs to monitor and modify WebLogic Server resources.

- *SNMP Management Guide for Oracle WebLogic Server* describes using Simple Network Management Protocol (SNMP) to monitor WebLogic domains.

1.4 New and Changed WLST Features in This Release

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

2

WebLogic Server WLST Online and Offline Command Reference

This chapters lists and summarizes the WLST commands in alphabetical order indicates which commands can be used in offline mode, online mode, or both.

- Section 2.1, "WebLogic Server WLST Command Summary, Alphabetically By Command"
- Section 2.2, "WebLogic Server WLST Online Command Summary"
- Section 2.3, "WebLogic Server WLST Offline Command Summary"

> **Note:** You can list a summary of all online and offline commands from the command-line using the following commands, respectively:
>
> ```
> help("online")
> help("offline")
> ```

2.1 WebLogic Server WLST Command Summary, Alphabetically By Command

The following tables summarizes each of the WebLogic Server WLST commands, alphabetically by command.

Table 2–1   WebLogic Server WLST Command Summary

This command...	Enables you to...	Use with WLST...
activate	Activate changes saved during the current editing session but not yet deployed.	Online
addHelpCommand	Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the `help('commandGroup')` command.	Online or Offline
addHelpCommandGroup	Adds a new help command group to those shown by the WLST `help()` command.	Online or Offline
addListener	Add a JMX listener to the specified MBean.	Online
addTemplate	Extend the current WebLogic domain using an application or service extension template.	Offline
assign	Assign resources to one or more destinations.	Offline

Table 2–1 (Cont.) WebLogic Server WLST Command Summary

This command...	Enables you to...	Use with WLST...
cancelEdit	Cancel an edit session, release the edit lock, and discard all unsaved changes. This operation can be called by any user with administrator privileges, even if the user did not start the edit session.	Online
cd	Navigate the hierarchy of configuration or runtime beans.	Online or Offline
closeDomain	Close the current WebLogic domain.	Offline
closeTemplate	Close the current domain template.	Offline
configToScript	Convert an existing server configuration (`config` directory) to an executable WLST script.	Online or Offline
connect	Connect WLST to a WebLogic Server instance.	Online or Offline
create	Create a configuration bean of the specified type for the current bean.	Online or Offline
currentTree	Return the current location in the hierarchy.	Online
custom	Navigate to the root of custom MBeans that are registered in the Runtime MBean Server.	Online
delete	Delete an instance of a configuration bean of the specified type for the current configuration bean.	Online or Offline
deploy	Deploy an application to a WebLogic Server instance.	Online
disconnect	Disconnect WLST from a WebLogic Server instance.	Online
distributeApplication	Copy the deployment bundle to the specified targets.	Online
domainConfig	Navigate to the last MBean to which you navigated in the domain configuration hierarchy or to the root of the hierarchy, `DomainMBean`.	Online
domainCustom	Navigate to the tree of custom MBeans that are registered in the Domain Runtime MBean Server.	Online
domainRuntime	Navigate to the last MBean to which you navigated in the domain runtime hierarchy or to the root of the hierarchy, `DomainRuntimeMBean`.	Online
dumpStack	Display stack trace from the last exception that occurred while performing a WLST action, and reset the stack trace.	Online or Offline
dumpVariables	Display all variables used by WLST, including their name and value.	Online or Offline
edit	Navigate to the last MBean to which you navigated in the configuration edit MBean hierarchy or to the root of the hierarchy, `DomainMBean`.	Online
encrypt	Encrypt the specified string.	Online
exit	Exit WLST from the user session and close the scripting shell.	Online or Offline

Table 2–1 (Cont.) WebLogic Server WLST Command Summary

This command...	Enables you to...	Use with WLST...
exportDiagnosticData	Execute a query against the specified log file.	Offline
exportDiagnosticDataFromServer	Executes a query on the server side and retrieves the exported WebLogic Diagnostic Framework (WLDF) data.	Online
find	Find MBeans and attributes in the current hierarchy.	Online
get	Return the value of the specified attribute.	Online or Offline
getActivationTask	Return the latest `ActivationTask` MBean on which a user can get status.	Online
getAvailableCapturedImages	Returns a list of the previously captured diagnostic images.	Online
getConfigManager	Return the latest `ConfigurationManagerBean` MBean which manages the change process.	Online
getMBean	Return the MBean by browsing to the specified path.	Online
getMBI	Return the `MBeanInfo` for the specified `MBeanType` or the `cmo` variable.	Online
getPath	Return the MBean path for the specified MBean instance.	Online
getWLDM	Return the WebLogic `DeploymentManager` object.	Online
invoke	Invoke a management operation on the current configuration bean.	Online
isRestartRequired	Determine whether a server restart is required.	Online
jndi	Navigates to the JNDI tree for the server to which WLST is currently connected.	Online
listApplications	List all applications that are currently deployed in the domain.	Online
listChildTypes	List all the children MBeans that can be created or deleted for the `cmo`.	Online
loadApplication	Load an application and deployment plan into memory.	Online or Offline
loadDB	Load SQL files into a database.	Offline
loadProperties	Load property values from a file.	Online and Offline
lookup	Look up the specified MBean.	Online
ls	List all child beans and/or attributes for the current configuration or runtime bean.	Online or Offline
man	Display help from `MBeanInfo` for the current MBean or its specified attribute.	Online
migrate	Migrate services to a target server within a cluster.	Online
nm	Determine whether WLST is connected to Node Manager.	Online
nmConnect	Connect WLST to Node Manager to establish a session.	Online or Offline

Table 2–1 (Cont.) WebLogic Server WLST Command Summary

This command...	Enables you to...	Use with WLST...
nmDisconnect	Disconnect WLST from a Node Manager session.	Online or Offline
nmEnroll	Enroll the machine on which WLST is currently running.	Online
nmGenBootStartupProps	Generates the Node Manager property files, `boot.properties` and `startup.properties`, for the specified server.	Online
nmKill	Kill the specified server instance that was started with Node Manager.	Online or Offline
nmLog	Return the Node Manager log.	Online or Offline
nmServerLog	Return the server output log of the server that was started with Node Manager.	Online or Offline
nmServerStatus	Return the status of the server that was started with Node Manager.	Online or Offline
nmStart	Start a server in the current domain using Node Manager.	Online or Offline
nmVersion	Return the Node Manager server version.	Online or Offline
prompt	Toggle the display of path information at the prompt.	Online or Offline
pwd	Display the current location in the configuration or runtime bean hierarchy.	Online or Offline
readDomain	Open an existing WebLogic domain for updating.	Offline
readTemplate	Open an existing domain template for WebLogic domain creation.	Offline
redeploy	Reload classes and redeploy a previously deployed application.	Online
redirect	Redirect WLST output to the specified filename.	Online or Offline
removeListener	Remove a listener that was previously defined.	Online
resume	Resume a server instance that is suspended or in `ADMIN` state.	Online
save	Save the edits that have been made but have not yet been saved.	Online
saveDiagnosticImageCaptureFile	Downloads the specified diagnostic image capture.	Online
saveDiagnosticImageCaptureEntryFile	Downloads a specific entry from the diagnostic image capture.	Online

Table 2–1 (Cont.) WebLogic Server WLST Command Summary

This command...	Enables you to...	Use with WLST...
serverRuntime	Navigate to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, ServerRuntimeMBean.	Online
set	Set the specified attribute value for the current configuration bean.	Online or Offline
setOption	Set options related to a WebLogic domain creation or update	Offline
showChanges	Show the changes made by the current user during the current edit session.	Online
showListeners	Show all listeners that are currently defined.	Online
shutdown	Gracefully shut down a running server instance or cluster.	Online
start	Start a Managed Server instance or a cluster using Node Manager.	Online
startApplication	Start an application, making it available to users.	Online
startEdit	Start a configuration edit session on behalf of the currently connected user.	Online
startNodeManager	Start Node Manager at default port (5556).	Online or Offline
startRecording	Record all user interactions with WLST; useful for capturing commands to replay.	Online or Offline
startServer	Start the Administration Server.	Online or Offline
state	Returns a map of servers or clusters and their state using Node Manager.	Online
stopApplication	Stop an application, making it un available to users.	Online
stopEdit	Stop the current edit session, release the edit lock, and discard unsaved changes.	Online
stopNodeManager	Stop Node Manager.	Online or Offline
stopRecording	Stop recording WLST commands.	Online or Offline
stopRedirect	Stop the redirection of WLST output to a file.	Online or Offline
storeUserConfig	Create a user configuration file and an associated key file.	Online
suspend	Suspend a running server.	Online
threadDump	Display a thread dump for the specified server.	Online or Offline
undeploy	Undeploy an application from the specified servers.	Online

Table 2–1 (Cont.) WebLogic Server WLST Command Summary

This command...	Enables you to...	Use with WLST...
updateApplication	Update an application configuration using a new deployment plan.	Online
updateDomain	Update and save the current domain.	Offline
unassign	Unassign applications or services from one or more destinations.	Offline
undo	Revert all unsaved or unactivated edits.	Online
validate	Validate the changes that have been made but have not yet been saved.	Online
viewMBean	Display information about an MBean, such as the attribute names and values, and operations.	Online
writeDomain	Write the domain configuration information to the specified directory.	Offline
writeIniFile	Convert WLST definitions and method declarations to a Python (.py) file.	Online or Offline
writeTemplate	Writes the domain configuration information to the specified domain template.	Offline

2.2 WebLogic Server WLST Online Command Summary

The following table summarizes the WebLogic Server WLST online commands, alphabetically by command.

Table 2–2 WebLogic Server WLST Online Command Summary

This command...	Enables you to...
activate	Activate changes saved during the current editing session but not yet deployed.
addHelpCommand	Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the help('commandGroup') command.
addHelpCommandGroup	Adds a new help command group to those shown by the WLST help() command, and specifies the resource bundle in which the help information is defined for the group.
addListener	Add a JMX listener to the specified MBean.
cancelEdit	Cancel an edit session, release the edit lock, and discard all unsaved changes. This operation can be called by any user with administrator privileges, even if the user did not start the edit session.
cd	Navigate the hierarchy of configuration or runtime beans.
configToScript	Convert an existing server configuration (config directory) to an executable WLST script.
connect	Connect WLST to a WebLogic Server instance.
create	Create a configuration bean of the specified type for the current bean.

Table 2–2   (Cont.) WebLogic Server WLST Online Command Summary

This command...	Enables you to...
currentTree	Return the current tree location.
custom	Navigate to the root of custom MBeans that are registered in the Runtime MBean Server.
delete	Delete an instance of a configuration bean of the specified type for the current configuration bean.
deploy	Deploy an application to a WebLogic Server instance.
disconnect	Disconnect WLST from a WebLogic Server instance.
distributeApplication	Copy the deployment bundle to the specified targets.
domainConfig	Navigate to the last MBean to which you navigated in the domain configuration hierarchy or to the root of the hierarchy, `DomainMBean`.
domainCustom	Navigate to the tree of custom MBeans that are registered in the Domain Runtime MBean Server.
domainRuntime	Navigate to the last MBean to which you navigated in the domain runtime hierarchy or to the root of the hierarchy, `DomainRuntimeMBean`.
dumpStack	Display stack trace from the last exception that occurred, and reset the trace.
dumpVariables	Display all variables used by WLST, including their name and value.
edit	Navigate to the last MBean to which you navigated in the configuration edit MBean hierarchy or to the root of the hierarchy, `DomainMBean`.
encrypt	Encrypt the specified string.
exit	Exit WLST from the interactive session and close the scripting shell.
exportDiagnosticDataFromServer	Execute a query on the server side and retrieves the exported WebLogic Diagnostic Framework (WLDF) data.
find	Find MBeans and attributes in the current hierarchy.
get	Return the value of the specified attribute.
getActivationTask	Return the latest `ActivationTask` MBean on which a user can get status.
getAvailableCapturedImages	Returns a list of the previously captured diagnostic images.
getConfigManager	Return the latest `ConfigurationManagerBean` MBean which manages the change process.
getMBean	Return the MBean by browsing to the specified path.
getMBI	Return the `MBeanInfo` for the specified `MBeanType` or the `cmo` variable.
getPath	Return the MBean path for the specified MBean instance.
getWLDM	Return the WebLogic `DeploymentManager` object.
invoke	Invoke a management operation on the current configuration bean.
isRestartRequired	Determine whether a server restart is required.

Table 2–2   (Cont.) WebLogic Server WLST Online Command Summary

This command...	Enables you to...
jndi	Navigates to the JNDI tree for the server to which WLST is currently connected.
listApplications	List all applications that are currently deployed in the domain.
listChildTypes	List all the children MBeans that can be created or deleted for the cmo.
loadApplication	Load an application and deployment plan into memory.
loadProperties	Load property values from a file.
lookup	Look up the specified MBean.
ls	List all child beans and/or attributes for the current configuration or runtime bean.
man	Display help from MBeanInfo for the current MBean or its specified attribute.
migrate	Migrate services to a target server within a cluster.
nm	Determine whether WLST is connected to Node Manager.
nmConnect	Connect WLST to Node Manager to establish a session.
nmDisconnect	Disconnect WLST from a Node Manager session.
nmEnroll	Enroll the machine on which WLST is currently running.
nmGenBootStartupProps	Generates the Node Manager property files, boot.properties and startup.properties, for the specified server.
nmKill	Kill the specified server instance that was started with Node Manager.
nmLog	Return the Node Manager log.
nmServerLog	Return the server output log of the server that was started with Node Manager.
nmServerStatus	Return the status of the server that was started with Node Manager.
nmStart	Start a server in the current domain using Node Manager.
nmVersion	Return the Node Manager server version.
prompt	Toggle the display of path information at the prompt.
pwd	Display the current location in the configuration or runtime bean hierarchy.
redeploy	Reload classes and redeploy a previously deployed application.
redirect	Redirect WLST output to the specified filename.
removeListener	Remove a listener that was previously defined.
resume	Resume a server instance that is suspended or in ADMIN state.
save	Save the edits that have been made but have not yet been saved.
saveDiagnosticImageCaptureFile	Downloads the specified diagnostic image capture.
saveDiagnosticImageCaptureEntryFile	Downloads a specific entry from the diagnostic image capture.

Table 2–2 (Cont.) WebLogic Server WLST Online Command Summary

This command...	Enables you to...
serverConfig	Navigate to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, `DomainMBean`.
serverRuntime	Navigate to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, `ServerRuntimeMBean`.
set	Set the specified attribute value for the current configuration bean.
showChanges	Show the changes made by the current user during the current edit session.
showListeners	Show all listeners that are currently defined.
shutdown	Gracefully shut down a running server instance or cluster.
start	Start a Managed Server instance or a cluster using Node Manager.
startApplication	Start an application, making it available to users.
startEdit	Start a configuration edit session on behalf of the currently connected user.
startNodeManager	Start Node Manager at default port (5556).
startRecording	Record all user interactions with WLST; useful for capturing commands to replay.
startServer	Start the Administration Server.
state	Returns a map of servers or clusters and their state using Node Manager
stopApplication	Stop an application, making it un available to users.
stopEdit	Stop the current edit session, release the edit lock, and discard unsaved changes.
stopNodeManager	Stop Node Manager.
stopRedirect	Stop the redirection of WLST output to a file.
storeUserConfig	Create a user configuration file and an associated key file.
suspend	Suspend a running server.
threadDump	Display a thread dump for the specified server.
undeploy	Undeploy an application from the specified servers.
undo	Revert all unsaved or unactivated edits.
updateApplication	Update an application configuration using a new deployment plan.
validate	Validate the changes that have been made but have not yet been saved.
viewMBean	Display information about an MBean, such as the attribute names and values, and operations.
writeIniFile	Convert WLST definitions and method declarations to a Python (`.py`) file.

2.3 WebLogic Server WLST Offline Command Summary

The following table summarizes the WebLogic Server WLST offline commands, alphabetically by command.

Table 2–3   WebLogic Server WLST Offline Command Summary

This command...	Enables you to...
addHelpCommand	Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the `help('commandGroup')` command.
addHelpCommandGroup	Adds a new help command group to those shown by the WLST `help()` command, and specifies the resource bundle in which the help information is defined for the group.
addTemplate	Extend the current domain using an application or service extension template.
assign	Assign resources to one or more destinations.
cd	Navigate the hierarchy of configuration or runtime beans.
closeDomain	Close the current domain.
closeTemplate	Close the current domain template.
configToScript	Convert an existing server configuration (`config` directory) to an executable WLST script.
connect	Connect WLST to a WebLogic Server instance.
create	Create a configuration bean of the specified type for the current bean.
delete	Delete an instance of a configuration bean of the specified type for the current configuration bean.
dumpStack	Display stack trace from the last exception that occurred while performing a WLST action, and reset the stack trace.
dumpVariables	Display all variables used by WLST, including their name and value.
exit	Exit WLST from the interactive session and close the scripting shell.
exportDiagnosticData	Execute a query against the specified log file.
get	Return the value of the specified attribute.
loadApplication	Load an application and deployment plan into memory.
loadDB	Load SQL files into a database.
loadProperties	Load property values from a file.
ls	List all child beans and/or attributes for the current configuration or runtime bean.
nmConnect	Connect WLST to Node Manager to establish a session.
prompt	Toggle the display of path information at the prompt.
pwd	Display the current location in the configuration or runtime bean hierarchy.
readDomain	Open an existing WebLogic domain for updating.
readTemplate	Open an existing domain template for domain creation.

Table 2–3   (Cont.) WebLogic Server WLST Offline Command Summary

This command...	Enables you to...
redirect	Redirect WLST output to the specified filename.
set	Set the specified attribute value for the current configuration bean.
setOption	Set options related to a WebLogic domain creation or update.
startNodeManager	Start Node Manager at default port (5556).
startRecording	Record all user interactions with WLST; useful for capturing commands to replay.
startServer	Start the Administration Server.
stopNodeManager	Stop Node Manager.
stopRedirect	Stop the redirection of WLST output to a file.
threadDump	Display a thread dump for the specified server.
unassign	Unassign applications or services from one or more destinations.
updateDomain	Update and save the current domain.
writeDomain	Write the domain configuration information to the specified directory.
writeIniFile	Convert WLST definitions and method declarations to a Python (`.py`) file.
writeTemplate	Writes the domain configuration information to the specified domain template.

3

WLST Command and Variable Reference

This chapter provides and overview of WLST command categories, and describes the WLST commands in detail. It also contains a WLST variable reference. It includes the following sections:

3.1 Overview of WLST Command Categories

> **Note:** Refer to "Syntax for WLST Commands" in *Oracle WebLogic Scripting Tool* for command syntax requirements.

WLST commands are divided into the following categories.

Table 3–1   WLST Command Categories

Command Category	Description
Section 3.2, "Browse Commands"	Navigate the hierarchy of configuration or runtime beans and control the prompt display.
Section 3.3, "Control Commands"	- Connect to or disconnect from a server. - Create and configure a WebLogic domain or domain template. - Exit WLST.

Table 3–1 (Cont.) WLST Command Categories

Command Category	Description
Section 3.4, "Customization Commands"	Add the command group help and command help that is displayed by the WLST help() and help('*commandGroup*') commands.
Section 3.5, "Deployment Commands"	■ Deploy, undeploy, and redeploy applications and standalone modules to a WebLogic Server instance. ■ Update an existing deployment plan. ■ Interrogate the WebLogic Deployment Manager object. ■ Start and stop a deployed application.
Section 3.6, "Diagnostics Commands"	Export diagnostic data.
Section 3.7, "Editing Commands"	Interrogate and edit configuration beans.
Section 3.8, "Information Commands"	Interrogate WebLogic domains, servers, and variables, and provide configuration bean, runtime bean, and WLST-related information.
Section 3.9, "Life Cycle Commands"	Manage the life cycle of a server instance.
Section 3.10, "Node Manager Commands"	Start, shut down, restart, and monitor WebLogic Server instances using Node Manager.
Section 3.11, "Tree Commands"	Navigate among MBean hierarchies.

3.2 Browse Commands

Use the WLST browse commands, listed in Table 3–2, to navigate the hierarchy of configuration or runtime beans and control the prompt display.

Table 3–2 Browse Commands for WLST Configuration

Use this command...	To...	Use with WLST...
cd	Navigate the hierarchy of configuration or runtime beans.	Online or Offline
currentTree	Return the current location in the hierarchy.	Online
prompt	Toggle the display of path information at the prompt.	Online or Offline
pwd	Display the current location in the hierarchy.	Online or Offline

3.2.1 cd

Command Category: Browse Commands

Use with WLST: Online or Offline

3.2.1.1 Description

Navigates the hierarchy of configuration or runtime beans. This command uses a model that is similar to navigating a file system in a Windows or UNIX command shell. For example, to navigate back to a parent configuration or runtime bean, enter cd('..'). The character string .. (dot-dot), refers to the directory immediately

above the current directory. To get back to the root bean after navigating to a bean that is deep in the hierarchy, enter cd('/').

You can navigate to beans in the current hierarchy and to any child or instance.

The cd command returns a stub of the configuration or runtime bean instance, if one exists. If you navigate to a type, this command returns a stub of the configuration or runtime bean instance from which you navigated. In the event of an error, the command returns a WLSTException.

> **Note:** The cmo variable is initialized to the root of all domain configuration beans when you first connect WLST to a server instance. It reflects the parent configuration bean type until you navigate to an instance. For more information about the cmo variable, see "Changing the Current Management Object" in *Oracle WebLogic Scripting Tool*.

3.2.1.2 Syntax

cd(mbeanName)

Argument	Definition
mbeanName	Path to the bean in the namespace.

3.2.1.3 Examples

The following example navigates the hierarchy of configuration beans. The first command navigates to the Servers configuration bean type, the second, to the myserver configuration bean instance, and the last back up two levels to the original directory location.

```
wls:/mydomain/serverConfig> cd('Servers')
wls:/mydomain/serverConfig/Servers> cd('myserver')
wls:/mydomain/serverConfig/Servers/myserver> cd('../..')
wls:/mydomain/serverConfig>
```

3.2.2 currentTree

Command Category: Browse Commands

Use with WLST: Online

3.2.2.1 Description

Returns the current location in the hierarchy. This command enables you to store the current location in the hierarchy and easily return to it after browsing. In the event of an error, the command returns a WLSTException.

3.2.2.2 Syntax

currentTree()

3.2.2.3 Example

The following example stores the current location in the hierarchy in myTree and uses it to navigate back to the Edit MBean hierarchy from the runtime MBean hierarchy on an Administration Server instance.

```
wls:/mydomain/edit> myTree=currentTree()
wls:/mydomain/edit> serverRuntime()
```

```
Location changed to serverRuntime tree. This is a read-only tree with
ServerRuntimeMBean as the root.
For more help, use help('serverRuntime')

wls:/mydomain/serverRuntime> myTree()
wls:/mydomain/edit>
```

3.2.3 prompt

Command Category: Browse Commands

Use with WLST: Online or Offline

3.2.3.1 Description

Toggles the display of path information at the prompt, when entered without an argument. This command is useful when the prompt becomes too long due to the length of the path.

You can also explicitly specify on or off as an argument to the command. When you specify off, WLST hides the WLST prompt and defaults to the Jython prompt. By default, the WLST prompt displays the configuration or runtime navigation path information.

When you disable the prompt details, to determine your current location in the hierarchy, you can use the pwd command, as described in Section 3.2.4, "pwd".

In the event of an error, the command returns a WLSTException.

3.2.3.2 Syntax

```
prompt(myPrompt)
```

Argument	Definition
myPrompt	Optional. Hides or displays WLST prompt. Valid values include off or on.
	■ The off argument hides the WLST prompt.
	If you run prompt('off'), when using WLST online, the prompt defaults to the Jython prompt. You can create a new prompt using Jython syntax. For more information about programming using Jython, see http://www.jython.org. In this case, if you subsequently enter the prompt command without arguments, WLST displays the WLST command prompt without the path information. To redisplay the path information, enter prompt() again, or enter prompt('on').
	■ The on argument displays the default WLST prompt, including the path information.

3.2.3.3 Examples

The following example hides and then redisplays the path information at the prompt.

```
wls:/mydomain/serverConfig/Servers/myserver> prompt()
wls:/> prompt()
wls:/mydomain/serverConfig/Servers/myserver>
```

The following example hides the prompt and defaults to the Jython prompt (since the command is run using WLST online), changes the Jython prompt, and then redisplays the WLST prompt. This example also demonstrates the use of the pwd command.

> **Note:** For more information about programming using Jython, see
> http://www.jython.org.

```
wls:/mydomain/serverConfig/Servers/myserver> prompt('off')
>>>sys.ps1="myprompt>"
myprompt> prompt()
wls:> pwd()
'serverConfig:Servers/myserver'
wls:> prompt()
wls:/mydomain/serverConfig/Servers/myserver>
```

3.2.4 pwd

Command Category: Browse Commands

Use with WLST: Online or Offline

3.2.4.1 Description

Displays the current location in the configuration or runtime bean hierarchy.

This command is useful when you have turned off the prompt display of the path information using the `prompt` command, as described in Section 3.2.3, "prompt".

In the event of an error, the command returns a `WLSTException`.

3.2.4.2 Syntax

```
pwd()
```

3.2.4.3 Example

The following example displays the current location in the configuration bean hierarchy.

```
wls:/mydomain/serverConfig/Servers/myserver/Log/myserver> pwd()
'serverConfig:/Servers/myserver/Log/myserver'
```

3.3 Control Commands

Use the WLST control commands, listed in Table 3–3, to perform the following tasks:

- Connect to or disconnect from a server (`connect` and `disconnect` commands)
- Create a new WebLogic domain from a domain template, similar to the Configuration Wizard (`createDomain`, `readTemplate`, `writeDomain`, and `closeTemplate` commands)
- Update an existing WebLogic domain, offline (`readDomain`, `addTemplate`, `updateDomain`, and `closeDomain` commands)
- Write a domain template (`writeTemplate` command)
- Exit WLST

Table 3–3 lists the control commands for WLST configuration.

Table 3–3   Control Commands for WLST Configuration

Use this command...	To...	Use with WLST...
connect	Connect WLST to a WebLogic Server instance.	Online or Offline
disconnect	Disconnect WLST from a WebLogic Server instance.	Online
createDomain	Create a new WebLogic domain using the specified template.	Offline
readTemplate	Open an existing domain template for domain creation.	Offline
writeDomain	Write the domain configuration information to the specified directory.	Offline
closeTemplate	Close the current domain template.	Offline
readDomain	Open an existing WebLogic domain for updating.	Offline
addTemplate	Extend the current WebLogic domain using an application or service extension template.	Offline
updateDomain	Update and save the current domain.	Offline
closeDomain	Close the current domain.	Offline
writeTemplate	Writes the configuration information to the specified domain template file.	Offline
exit	Exit WLST from the interactive session and close the scripting shell.	Online or Offline

3.3.1 addTemplate

Command Category: Control Commands

Use with WLST: Offline

3.3.1.1 Description

Extends the current WebLogic domain using an application or service extension template. Use the Template Builder to create an application or service extension template. See *Oracle WebLogic Server Creating Templates Using the Domain Template Builder*.

In the event of an error, the command returns a `WLSTException`.

3.3.1.2 Syntax

```
addTemplate(templateFileName)
```

Argument	Definition
templateFileName	Name of the application or service extension template.

3.3.1.3 Example

The following example opens a WebLogic domain and extends it using the specified extension template, `DefaultWebApp.jar`.

```
wls:/offline> readDomain('c:/Oracle/Middleware/user_projects/domains/wlw')
```

```
wls:/offline/wlw> addTemplate('c:/Oracle/Middleware/wlserver_12.1
/common/templates/applications/DefaultWebApp.jar')
wls:/offline/wlw>
```

3.3.2 closeDomain

Command Category: Control Commands

Use with WLST: Offline

3.3.2.1 Description

Closes the current domain. The domain is no longer available for editing once it is closed. In the event of an error, the command returns a `WLSTException`.

3.3.2.2 Syntax

```
closeDomain()
```

3.3.2.3 Example

The following example closes the current domain:

```
wls:/offline> readDomain('c:/Oracle/Middleware/user_projects/domains/medrec')
...
wls:/offline/medrec> updateDomain()
wls:/offline/medrec> closeDomain()
wls:/offline>
```

3.3.3 closeTemplate

Command Category: Control Commands

Use with WLST: Offline

3.3.3.1 Description

Closes the current domain template. The domain template is no longer available once it is closed. In the event of an error, the command returns a `WLSTException`.

3.3.3.2 Syntax

```
closeTemplate()
```

3.3.3.3 Example

The following example opens an existing domain template, performs some operations, and then closes the current domain template.

```
wls:/offline> readTemplate('c:/Oracle/Middleware/wlserver_12.1
/common/templates/domains/wls.jar')
...
wls:/offline/wls> closeTemplate()
wls:/offline>
```

3.3.4 connect

Command Category: Control Commands

Use with WLST: Online or Offline

3.3.4.1 Description

Connects WLST to a WebLogic Server instance.

Requires you to provide the credentials (user name and password) of a user who has been defined in the active WebLogic security realm. Once you are connected, a collection of security policies determine which configuration attributes you are permitted to view or modify. (See "Default Security Policies for MBeans" in the *WebLogic Server MBean Reference*.)

You can supply user credentials by doing any of the following:

- Enter the credentials on the command line. This option is recommended only if you are using WLST in interactive mode.

- Enter the credentials on the command line, then use the `storeUserConfig` command to create a user configuration file that contains your credentials in an encrypted form and a key file that WebLogic Server uses to unencrypt the credentials. On subsequent WLST sessions (or in WLST scripts), supply the name of the user configuration file and key file instead of entering the credentials on the command line. This option is recommended if you use WLST in script mode because it prevents you from storing unencrypted user credentials in your scripts.

- Use the credentials that are stored in the Administration Server's `boot.properties` file. By default, when you create an Administration Server in development mode, WebLogic Server encrypts the credentials that were used to create the server and stores them in a `boot.properties` file. When you create an Administration Server in production mode, no `boot.properties` file is created. If your production domain does not contain a `boot.properties` file, you can create one manually; see "Creating a Boot Identify File for an Administration Server" in *Managing Server Startup and Shutdown for Oracle WebLogic Server*.

    When you run the `connect` command, if there is a `boot.properties` file containing the encrypted username and password for the domain, you do not have to enter the username and password to connect to the Administration Server. You do, however, have to specify the name of the Administration Server in the `connect` command.

Please note:

- If you run the `connect` command in a script without specifying the username and password or user configuration file and key file, a `WSLTException` occurs. In interactive mode, you are prompted for the username and password.

- Oracle strongly recommends that you connect WLST to the server through the SSL port or administration port. If you do not, the following warning message is displayed:

    ```
    Warning: An insecure protocol was used to connect to the server. To ensure
    on-the-wire security, the SSL port or Admin port should be used instead.
    ```

- If you are connecting to a WebLogic Server instance through an SSL listen port on a server that is using the demonstration SSL keys and certificates, invoke WLST using the following command:

    ```
    java -Dweblogic.security.SSL.ignoreHostnameVerification=true
    -Dweblogic.security.TrustKeyStore=DemoTrust weblogic.WLST
    ```

    For more information about invoking WLST, see "Main Steps for Using WLST in Interactive or Script Mode" in *Oracle WebLogic Scripting Tool*.

- If you are connecting to a WebLogic Server instance via HTTP, ensure that the
  `TunnelingEnabled` attribute is set to `true` for the WebLogic Server instance.
  For more information, see "TunnelingEnabled" in *Oracle WebLogic Server MBean
  Reference*.

- When trying to connect to the WebLogic Server Administration Server from WLST
  using localhost as the host name, the following message may be displayed if the
  listen-address attribute of the Administration Server has been restricted to certain
  IP addresses:

```
javax.naming.CommunicationException [Root exception is
java.net.ConnectException : <t3://HOST:PORT> : Destination unreachable;
nested exception is: java.net.ConnectException: Connection refused; No
available router to destination
```

You can use either of the following workarounds for this issue:

- Check that the listen-address attribute of the Administration Server has been
  set correctly. For example, in the domain configuration file:

```
<server>
    <name>AdminServer</name>
    <ssl>
    .
    .
    .
    </ssl>
    <machine>your_machine</machine>
    <!-- listen-address><your_ip_address></listen-address -->
</server>
```

- Use the hostname of the Administration Server, instead of localhost, in the
  WLST connect command.

After successfully connecting to a WebLogic Server instance, all the local variables are
initialized.

In the event of an error, the command returns a `WLSTException`.

3.3.4.2 Syntax

```
connect([username, password], [url], [timeout])
connect([userConfigFile, userKeyFile], [url], [timeout])
connect([url], [adminServerName], [timeout])
```

Argument	Definition
username	Optional. Username of the operator who is connecting WLST to the server. If not specified, WLST processes the command as described above.
password	Optional. Password of the operator who is connecting WLST to the server. If not specified, WLST processes the command as described above.
url	Optional. Listen address and listen port of the server instance, specified using the following format: [*protocol*://]*listen-address:listen-port*. If not specified, this argument defaults to `t3://localhost:7001`.

Argument	Definition
timeout	Optional. The number of milliseconds that WLST waits for online commands to complete (return).
	When you invoke a WLST online command, WLST connects to an MBean Server, invokes an MBean server method, and returns the results of the invocation. If the MBean server method does not return within the timeout period, WLST abandons its invocation attempt. Use the following syntax for this argument:
	`timeout='milliseconds'`
	A value of `0` indicates that the operation will not time out. This argument defaults to 300,000 ms (or 5 minutes).
userConfigFile	Optional. Name and location of a user configuration file which contains an encrypted username and password. Use the following syntax for this argument:
	`userConfigFile='file-system-path'`
	If not specified, WLST processes the command as described above.
	When you create a user configuration file, the `storeUserConfig` command uses a key file to encrypt the username and password. Only the key file that encrypts a user configuration file can decrypt the username and password. (See Section 3.8.21, "storeUserConfig".)
userKeyFile	Optional. Name and location of the key file that is associated with the specified user configuration file and is used to decrypt it. Use the following syntax for this argument:
	`userKeyFile='file-system-path'`
	If not specified, WLST processes the command as described above.
	See Section 3.8.21, "storeUserConfig".
adminServerName	Optional. Name of the Administration Server for the domain. Causes the connect command to use the credentials that are stored in the Administration Server's `boot.properties` file. Use the following syntax for this argument:
	`adminServerName='server-name'`
	This argument is valid only when you start WLST from a domain directory. If the `boot.properties` file for the Administration Server is located in the domain directory, then you do not need to specify this argument.
	If not specified, WLST processes the command as described above.

3.3.4.3 Examples

The following example connects WLST to a WebLogic Server instance. In this example, the Administration Server name defaults to `AdminServer`. Note that a warning is displayed if the SSL or administration port is not used to connect to the server.

```
wls:/offline> connect('weblogic','welcome1','t3://localhost:8001')
Connecting to weblogic server instance running at t3://localhost:8001 as
username weblogic...

Successfully connected to Admin Server 'AdminServer' that belongs to domain
'mydomain'.

Warning: An insecure protocol was used to connect to the server. To ensure
on-the-wire security, the SSL port or Admin port should be used instead.

wls:/mydomain/serverConfig>
```

The following example connects WLST to a WebLogic Server instance at the specified URL. In this example, the username and password are passed as variables. This example uses a secure protocol.

```
wls:/offline> username = 'weblogic'
wls:/offline> password = 'welcome1'
wls:/offline> connect(username,password,'t3s://myhost:8001')
Connecting to weblogic server instance running at t3://myhost:8001 as
username weblogic...

Successfully connected to Admin Server 'AdminServer' that belongs to domain
'mydomain'.
wls:/mydomain/serverConfig>
```

The following example connects WLST to a WebLogic Server instance using a user configuration and key file to provide user credentials.

```
wls:/offline> connect(userConfigFile='c:/myfiles/myuserconfigfile.secure',
userKeyFile='c:/myfiles/myuserkeyfile.secure')
Connecting to t3://localhost:7001 with userid username ...

Successfully connected to Admin Server 'AdminServer' that belongs to domain
'mydomain'.
wls:/mydomain/serverConfig>
```

The following example shows the prompts that are displayed in interactive mode if you run the command without parameters:

```
wls:/offline> connect()
Please enter your username :username
Please enter your password :
Please enter your server URL [t3://localhost:7001] :
Connecting to t3//localhost:7001 with userid username
```

3.3.5 createDomain

Command Category: Control Commands

Use with WLST: Offline

3.3.5.1 Description

Creates a WebLogic domain using the specified template.

> **Note:**   If you wish to modify the domain configuration settings when creating a WebLogic domain, see Option 2 in "Editing a Domain (Offline)" in *Oracle WebLogic Scripting Tool*.
>
> The createDomain command is similar in functionality to the unpack command, as described in *Creating Templates and Domains Using the pack and unpack Commands*.

In the event of an error, the command returns a WLSTException.

3.3.5.2 Syntax

```
createDomain(domainTemplate, domainDir, user, password)
```

Argument	Definition
domainTemplate	Name and location of the domain template from which you want to create a domain.
domainDir	Name of the directory to which you want to write the domain configuration information.
	Oracle recommends that you create all domains for your environment outside of the Middleware home directory. This makes it easier for you to remove an existing installation or install a new version of WebLogic Server without having to recreate your domains and applications.
user	Name of the default user.
password	Password of the default user.

3.3.5.3 Example

The following example creates a new WebLogic domain using the Avitek MedRec template and sets the default username to `weblogic` and the password to `welome1`. The domain is saved to the following directory: `c:/Oracle/Middleware/wlserver_12.1/user_projects/domains/medrec`.

```
wls:/offline> createDomain('c:/Oracle/Middleware/wlserver_12.1/common
/templates/domains/wls_medrec.jar','c:/Oracle/Middleware/user_
projects/domains/medrec',
'weblogic', 'welcome1')
```

3.3.6 disconnect

Command Category: Control Commands

Use with WLST: Online

3.3.6.1 Description

Disconnects WLST from a WebLogic Server instance. The `disconnect` command does not cause WLST to exit the interactive scripting shell; it closes the current WebLogic Server instance connection and resets all the variables while keeping the interactive shell alive.

In the event of an error, the command returns a `WLSTException`.

You can connect to another WebLogic Server instance using the `connect` command, as described in Section 3.3.4, "connect".

3.3.6.2 Syntax

```
disconnect(force)
```

Argument	Definition
force	Optional. Boolean value specifying whether WLST should disconnect without waiting for the active sessions to complete. This argument defaults to `false`, indicating that all active sessions must complete before disconnect.

3.3.6.3 Example

The following example disconnects from a running server:

```
wls:/mydomain/serverConfig> disconnect()
```

```
Disconnected from weblogic server: myserver
wls:/offline>
```

3.3.7 exit

Command Category: Control Commands

Use with WLST: Online or Offline

3.3.7.1 Description

Exits WLST from the user session and closes the scripting shell.

If there is an edit session in progress, WLST prompts you for confirmation. To skip the prompt, set the *defaultAnswer* argument to y.

By default, WLST calls System.exit(0) for the current WLST JVM when exiting WLST. If you would like the JVM to exit with a different exit code, you can specify a value using the *exitCode* argument.

> **Note:** When the WLST exit command is issued within an Ant script, it may also exit the execution of the Ant script. It is recommended that when invoking WLST within an Ant script, you fork a new JVM by specifying fork="true".

In the event of an error, the command returns a WLSTException.

3.3.7.2 Syntax

exit([defaultAnswer], [exitcode])

Argument	Definition
defaultAnswer	Optional. Default response, if you would prefer not to be prompted at the command line. Valid values are y and n. This argument defaults to null, and WLST prompts you for a response.
exitcode	Optional. Exit code to set when exiting WLST.

3.3.7.3 Example

The following example disconnects from the user session and closes the scripting shell.

```
wls:/mydomain/serverConfig> exit()
Exiting WebLogic Scripting Tool ...
c:\>
```

The following example disconnects from the user session, closes the scripting shell, and sets the error code to 101.

```
wls:/mydomain/serverConfig> exit(exitcode=101)
Exiting WebLogic Scripting Tool ...
c:\>
```

3.3.8 readDomain

Command Category: Control Commands

Use with WLST: Offline

3.3.8.1 Description

Opens an existing WebLogic domain for updating.

WLST offline provides read and write access to the configuration data that is persisted in the `config` directory for the WebLogic domain, or in a domain template JAR created using Template Builder. This data is a collection of XML documents and expresses a hierarchy of management objects.

When you open a template or WebLogic domain, WLST is placed at the root of the configuration hierarchy for that domain, and the prompt is updated to reflect the current location in the configuration hierarchy. For example:

```
wls:/offline/base_domain>
```

For more information, see "Navigating and Interrogating MBeans" in *Oracle WebLogic Scripting Tool*.

In the event of an error, the command returns a `WLSTException`.

3.3.8.2 Syntax

```
readDomain(domainDirName)
```

Argument	Definition
domainDirName	Name of the WebLogic domain directory that you want to open.

3.3.8.3 Example

The following example opens the `medrec` domain for editing.

```
wls:/offline> readDomain('c:/Oracle/Middleware/user_projects/domains/medrec')
wls:/offline/medrec>
```

3.3.9 readTemplate

Command Category: Control Commands

Use with WLST: Offline

3.3.9.1 Description

Opens an existing domain template for domain creation.

When you open a domain template, WLST is placed into the configuration bean hierarchy for that domain template, and the prompt is updated to reflect the current location in the configuration hierarchy. For example:

```
wls:/offline/base_domain>
```

WebLogic Server configuration beans exist within a hierarchical structure. In the WLST file system, the hierarchies correspond to drives; types and instances are directories; attributes and operations are files. WLST traverses the hierarchical structure of configuration beans using commands such as cd, ls, and pwd in a similar way that you would navigate a file system in a UNIX or Windows command shell. After navigating to a configuration bean instance, you interact with the bean using WLST commands. For more information, see "Navigating and Interrogating MBeans" in *Oracle WebLogic Scripting Tool*.

> **Note:** Using WLST and a domain template, you can only create and access security information when you are creating a new WebLogic domain. When you are updating a WebLogic domain, you cannot access security information through WLST.

In the event of an error, the command returns a `WLSTException`.

3.3.9.2 Syntax

```
readTemplate(templateFileName)
```

Argument	Definition
templateFileName	Name of the JAR file corresponding to the domain template.

3.3.9.3 Example

The following example opens the `medrec.jar` domain template for WebLogic domain creation.

```
wls:/offline> readTemplate('c:/Oracle/Middleware/wlserver_12.1/common/templates
/domains/wls_medrec.jar')
wls:/offline/wls_medrec>
```

3.3.10 updateDomain

Command Category: Control Commands

Use with WLST: Offline

3.3.10.1 Description

Updates and saves the current WebLogic domain. The domain continues to be editable after you update and save it.

In the event of an error, the command returns a `WLSTException`.

3.3.10.2 Syntax

```
updateDomain()
```

3.3.10.3 Example

The following examples opens the medrec domain, performs some operations, and updates and saves the current domain:

```
wls:/offline> readDomain('c:/Oracle/Middleware/user_projects/domains/medrec')
...
wls:/offline/medrec> updateDomain()
```

3.3.11 writeDomain

Command Category: Control Commands

Use with WLST: Offline

3.3.11.1 Description

Writes the domain configuration information to the specified directory.

Once you write the WebLogic domain to file system, you can continue to update the domain template object that exists in memory, and reissue the `writeDomain` command to store the domain configuration to a new or existing file.

By default, when you write a WebLogic domain, the associated applications are written to *WL_HOME*/user_projects/applications/*domainname*, where *WL_HOME* specifies the WebLogic Server home directory and *domainname* specifies the name of the WebLogic domain. This directory must be empty; otherwise, WLST displays an error.

When you have finished using the domain template object in memory, close it using the `closeTemplate` command. If you want to edit the WebLogic domain that has been saved to disk, you can open it using the `readDomain` command.

> **Note:** The name of the WebLogic domain is derived from the name of the domain directory. For example, for a domain saved to `c:/Oracle/Middleware/user_projects/domains/myMedrec`, the domain name is `myMedrec`.

Before writing the domain, you must define a password for the default user, if it is not already defined. For example:

```
cd('/Security/base_domain/User/weblogic')
cmo.setPassword('welcome1')
```

In the event of an error, the command returns a `WLSTException`.

3.3.11.2 Syntax

```
writeDomain(domainDir)
```

Argument	Definition
domainDir	Name of the directory to which you want to write the domain configuration information.

3.3.11.3 Example

The following example reads the medrec.jar domain templates, performs some operations, and writes the domain configuration information to the `c:/Oracle/Middleware/user_projects/domains/medrec` directory.

```
wls:/offline> readTemplate('c:/Oracle/Middleware/wlserver_12.1/common/templates
/domains/wls.jar')
...
wls:/offline/base_domain> writeDomain('c:/Oracle/Middleware/user_
projects/domains/base_domain')
```

3.3.12 writeTemplate

Command Category: Control Commands

Use with WLST: Offline

3.3.12.1 Description

Writes the domain configuration information to the specified domain template. You can use the domain configuration template to recreate the WebLogic domain.

Once your write the configuration information to the domain configuration template, you can continue to update the WebLogic domain or domain template object that exists in memory, and reissue the writeDomain or writeTemplate command to store the domain configuration to a new or existing WebLogic domain or domain template file. For more information, see Section 3.3.11, "writeDomain" or Section 3.3.12, "writeTemplate", respectively.

In the event of an error, the command returns a WLSTException.

> **Note:** The writeTemplate command is similar in functionality to the pack command; see "The pack Command" in *Creating Templates and Domains Using the pack and unpack Commands*. However, writeTemplate does not support creating a Managed Server template.

3.3.12.2 Syntax

writeTemplate(templateName)

Argument	Definition
templateName	Name of the domain template to store the domain configuration information.

3.3.12.3 Example

The following example writes the current domain configuration to the domain template named c:/Oracle/Middleware/user_projects/templates/myTemplate.jar.

```
wls:/offline> readDomain('c:/Oracle/Middleware/user_projects/domains/mydomain')
...
wls:/offline/base_domain> writeTemplate('c:/Oracle/Middleware/user_projects
/templates/myTemplate.jar')
```

3.4 Customization Commands

Use the WLST customization commands, listed in Table 3–4, to add the command group help and command help that is listed by the WLST help() and help('*commandGroup*') commands. For more information about adding command help to WLST, see "Adding Integrated Help for Custom Commands" in *Oracle WebLogic Scripting Tool*.

Table 3–4   Customization Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
addHelpCommandGroup	Adds a new help command group to those shown by the WLST help() command.	Online or Offline
addHelpCommand	Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the help('commandGroup') command.	Online or Offline

3.4.1 addHelpCommandGroup

Command Category: Customization Commands

Use with WLST: Online or Offline

3.4.1.1 Description

Adds a new command help group to those shown by the WLST `help()` command, and specifies the resource bundle in which the help information is defined for the group.

3.4.1.2 Syntax

`addHelpCommandGroup(commandGroup, resourceBundleName)`

Argument	Definition
commandGroup	Use a unique name for the command group. Do not use a command group name that is already shown by the WLST `help()` command.
resourceBundleName	Represents either a class name or property resource file name. The resource bundle contains help text for entries for the command group using a standard pattern. The resource bundle name will be passed to `ResourceBundle.getBundle(...)`. Multiple command groups can use the same resource bundle.
	The resource bundle must be present in the classpath.
	See "Adding Integrated Help for Custom Commands" in *Oracle WebLogic Scripting Tool* for information on how to define the help text for each command group and command.
	For more information on resourceBundles and localization, refer to http://java.sun.com/javase/6/docs/api/java/util/ResourceBundle.html.

3.4.1.3 Examples

The following example adds the `boot` command group to the list of groups shown by the `help()` command, and specifies that the help text is located in the property resource file 'myhelp':

```
wls:/offline> addHelpCommandGroup('boot','myhelp')
```

The following example adds the `boot` command group to the list of groups shown by the `help()` command, and specifies that the help text is located in the class `foo.bar.MyResourceBundleClass`:

```
wls:/offline> addHelpCommandGroup('boot','foo.bar.MyResourceBundleClass')
```

3.4.2 addHelpCommand

Command Category: Customization Commands

Use with WLST: Online or Offline

3.4.2.1 Description

Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the `help('commandGroup')` command. You can also specify whether or not the command is listed by the `help('online')` and `help('offline')` commands.

3.4.2.2 Syntax

addHelpCommand(commandName,commandGroup,[offline=false, online=false])

Argument	Definition
commandName	The name of the command as defined in the command group specified by commandGroup.
commandGroup	The commandGroup to which the command belongs.
online	Optional. Boolean value that determines whether or not the command shows up in the help('online') output. The default value is 'false'.
offline	Optional. Boolean value that determines whether or not the command shows up in the help('offline') output. The default value is 'false'.

3.4.2.3 Example

The following example shows how to add the online command bootDB to the listing output by the help('boot') and help('online') commands:

wls:/offline> **addHelpCommand('bootDB','boot',online='true',offline='false')**

3.5 Deployment Commands

Use the WLST deployment commands, listed in Table 3–5, to:

- Deploy, undeploy, and redeploy applications and standalone modules to a WebLogic Server instance.

- Update an existing deployment plan.

- Interrogate the WebLogic Deployment Manager object.

- Start and stop a deployed application.

For more information about deploying applications, see *Deploying Applications to Oracle WebLogic Server*.

Table 3–5   Deployment Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
deploy	Deploy an application to a WebLogic Server instance.	Online
distributeApplication	Copy the deployment bundle to the specified targets.	Online
getWLDM	Return the WebLogic DeploymentManager object.	Online
listApplications	List all applications that are currently deployed in the WebLogic domain.	Online
loadApplication	Load an application and deployment plan into memory.	Online and Offline
redeploy	Redeploy a previously deployed application.	Online
startApplication	Start an application, making it available to users.	Online
stopApplication	Stop an application, making it unavailable to users.	Online

Table 3–5   (Cont.) Deployment Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
undeploy	Undeploy an application from the specified servers.	Online
updateApplication	Update an application configuration using a new deployment plan.	Online

3.5.1 deploy

Command Category: Deployment Commands

Use with WLST: Online

3.5.1.1 Description

Deploys an application to a WebLogic Server instance.

The `deploy` command returns a `WLSTProgress` object that you can access to check the status of the command. For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*. In the event of an error, the command returns a `WLSTException`.

> **Note:** If there is an edit session in progress, the `deploy` command does not block user interaction.

3.5.1.2 Syntax

```
deploy(appName, path, [targets], [stageMode], [planPath], [options])
```

Argument	Definition
appName	Name of the application or standalone Java EE module to be deployed.
path	Name of the application directory, archive file, or root of the exploded archive directory to be deployed.
targets	Optional. Comma-separated list of the targets. Each target may be qualified with a Java EE module name (for example, *module1@server1*) enabling you to deploy different modules of the application archive on different servers. This argument defaults to the server to which WLST is currently connected.
stageMode	Optional. Staging mode for the application you are deploying. Valid values are `stage`, `nostage`, and `external_stage`. For information about the staging modes, see "Controlling Deployment File Copying with Staging Modes" in *Deploying Applications to Oracle WebLogic Server*. If you do not specify a stage mode, the default stage mode is used. On the Administration Server, the default stage mode is `nostage` and on Managed Servers, it is `stage`.
planPath	Optional. Name of the deployment plan file. The filename can be absolute or relative to the application directory. This argument defaults to the `plan/plan.xml` file in the application directory, if one exists.

Argument	Definition
options	Optional. Comma-separated list of deployment options, specified as name-value pairs. Valid options include:

- **altDD**—Location of the alternate application deployment descriptor on the Administration Server.

- **altWlsDD**—Location of the alternate WebLogic application deployment descriptor on the Administration Server.

- **archiveVersion**—Archive version number.

- **block**—Boolean value specifying whether WLST should block user interaction until the command completes. This option defaults to `true`. If set to `false`, WLST returns control to the user after issuing the command; you can query the `WLSTProgress` object to determine the status of the command. If you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*, `block` is always set to `true`.

- **clusterDeploymentTimeout**—Time, in milliseconds, granted for a cluster deployment task on this application.

- **createPlan**—Boolean value indicating that user would like to create a default plan. This option defaults to `false`.

- **defaultSubmoduleTargets**—Boolean value indicating that targeting for qualifying JMS submodules should be derived by the system, see "Using Sub-Module Targeting with JMS Application Modules" in *Deploying Applications to Oracle WebLogic Server*. Default value is true.

- **deploymentPrincipalName**—String value specifying the principal for deploying the file or archive during server starts (static deployment; it does not effect the current deployment task). Make sure the user exists. This option adds `<deployment-principal-name>` to the `<app-deployment>` element in the `config.xml` file.

- **forceUndeployTimeout**—Force undeployment timeout value.

- **gracefulIgnoreSessions**—Boolean value specifying whether the graceful production to admin mode operation should ignore pending HTTP sessions. This option defaults to `false` and only applies if `gracefulProductionToAdmin` is set to `true`.

- **gracefulProductionToAdmin**—Boolean value specifying whether the production to Admin mode operation should be graceful. This option defaults to `false`.

- **libImplVersion**—Implementation version of the library, if it is not present in the manifest.

- **libraryModule**—Boolean value specifying whether the module is a library module. This option defaults to `false`.

Argument	Definition
options (Continued)	■ **libSpecVersion**—Specification version of the library, if it is not present in the manifest. ■ **planVersion**—Plan version number. ■ **remote**—Boolean value specifying whether the operation will be remote from the file system that contains the source. Use this option when you are on a different machine from the Administration Server and the deployment files are already at the specified location where the Administration Server is located. This option defaults to false. ■ **retireGracefully**—Retirement policy to gracefully retire an application only after it has completed all in-flight work. This policy is only meaningful for stop and redeploy operations and is mutually exclusive to the retire timeout policy. ■ **retireTimeout**—Time (in seconds) WLST waits before retiring an application that has been replaced with a newer version. This option default to -1, which specifies graceful timeout. ■ **securityModel**—Security model. Valid values include: DDOnly, CustomRoles, CustomRolesAndPolicies, and Advanced. ■ **securityValidationEnabled**—Boolean value specifying whether security validation is enabled. ■ **subModuleTargets**—Submodule level targets for JMS modules. For example, submod@mod-jms.xml@target | submoduleName@target. ■ **adminMode**—Boolean value specifying whether to start the Web application with restricted access. This option defaults to false. ■ **timeout**—Time (in milliseconds) that WLST waits for the deployment process to complete before canceling the operation. A value of 0 indicates that the operation will not time out. This argument defaults to 300,000 ms (or 5 minutes). ■ **upload**—Boolean value specifying whether the application files are uploaded to the WebLogic Server Administration Server's upload directory prior to deployment. Use this option when the Administration Server cannot access the application files through the file system. This option defaults to false. ■ **versionIdentifier**—Version identifier.

3.5.1.3 Example

The following example deploys the businessApp application located at c:/myapps/business, A default deployment plan is created.

The deploy command returns a WLSTProgress object that you can access to check the status of the command. The WLSTProgress object is captured in a user-defined variable, in this case, progress.

```
wls:/mydomain/serverConfig/Servers> progress= deploy(appName='businessApp',
path='c:/myapps/business',createplan='true')
```

The previous example stores the WLSTProgress object returned in a user-defined variable, in this case, progress. You can then use the progress variable to print the status of the deploy command. For example:

```
wls:/mydomain/serverConfig/Servers> progress.printStatus()
Current Status of your Deployment:
Deployment command type: deploy
Deployment State       : completed
Deployment Message     : null
```

```
wls:/mydomain/serverConfig/Servers>
```

For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*.

The following example deploys the `demoApp` application in the archive file located at `c:/myapps/demos/app/demoApp.ear`, targeting the application modules to `myserver`, and using the deployment plan file located in `c:/myapps/demos/app/plan/plan.xml`. WLST waits 120,000 ms for the process to complete.

```
wls:/mydomain/serverConfig/Servers> deploy('demoApp',
'c:/myapps/demos/app/demoApp.ear', targets='myserver',
planPath='c:/myapps/demos/app/plan/plan.xml', timeout=120000)
```

The following example deploys the `jmsApp` application located at `c:/myapps/demos/jmsApp/demo-jms.xml`, targeting the application module to a specific target.

```
wls:/mydomain/serverConfig/Servers> deploy('jmsApp',path=
'c:/myapps/demos/jmsApps/demo-jms.xml', subModuleTargets='jmsApp@managed1')
```

The following example shows how to set the application version (`appVersion`) to a unique identifier to support production (side-by-side) redeployment. This example deploys the `demoApp` application in the archive file located at `c:/myapps/demos/app/demoApp.ear`, and sets the application and archive version numbers to the specified values.

```
wls:/mydomain/serverConfig> deploy('demoApp', 'c:/myapps/demos/app/demoApp.ear',
archiveVersion='901-101', appVersion='901-102')
```

For more information about production redeployment strategies, see "Redeploying Applications in a Production Environment" in *Deploying Applications to Oracle WebLogic Server*.

3.5.2 distributeApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.2.1 Description

Copies the deployment bundle to the specified targets. The deployment bundle includes module, configuration data, and any additional generated code. The `distributeApplication` command does not start deployment.

The `distributeApplication` command returns a `WLSTProgress` object that you can access to check the status of the command. For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*. In the event of an error, the command returns a `WLSTException`.

3.5.2.2 Syntax

distributeApplication(appPath, [planPath], [targets], [options])

Argument	Definition
appPath	Name of the archive file or root of the exploded archive directory to be deployed.

Argument	Definition
planPath	Optional. Name of the deployment plan file. The filename can be absolute or relative to the application directory. This argument defaults to the `plan/plan.xml` file in the application directory, if one exists.
targets	Optional. Comma-separated list of targets. Each target may be qualified with a Java EE module name (for example, *module1@server1*) enabling you to deploy different modules of the application archive on different servers. This argument defaults to the server to which WLST is currently connected.
options	Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see the *options* argument description in Section 3.5.1, "deploy".

3.5.2.3 Example

The following example loads the `BigApp` application located in the `c:/myapps` directory, and stores the `WLSTProgress` object in a user-defined variable, in this case, `progress`.

The following example distributes the `c:/myapps/BigApp` application to the `myserver`, `oamserver1`, and `oamcluster` servers, using the deployment plan defined at `c:/deployment/BigApp/plan.xml`.

```
wls:/offline> progress=distributeApplication('c:/myapps/BigApp',
'c:/deployment/BigApp/plan.xml', 'myserver,oamserver1,oamcluster')
Distributing Application and Plan ...
Successfully distributed the application.
```

The previous example stores the `WLSTProgress` object in a user-defined variable, in this case, `progress`. You can then use the `progress` variable to determine if the `distributeApplication` command has completed. For example:

```
wls:/mydomain/serverConfig/Servers> progress.isCompleted()
1
wls:/mydomain/serverConfig/Servers>
```

For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*.

3.5.3 getWLDM

Command Category: Deployment Commands

Use with WLST: Online

3.5.3.1 Description

Returns the WebLogic `DeploymentManager` object. You can use the object methods to configure and deploy applications. WLST must be connected to an Administration Server to run this command. In the event of an error, the command returns a `WLSTException`.

3.5.3.2 Syntax

```
getWLDM()
```

3.5.3.3 Example

The following example gets the `WebLogicDeploymentManager` object and stores it in the `wldm` variable.

```
wls:/mydomain/serverConfig> wldm=getWLDM()
wls:/mydomain/serverConfig> wldm.isConnected()
1
wls:/mydomain/serverConfig>
```

3.5.4 listApplications

Command Category: Deployment Commands

Use with WLST: Online

3.5.4.1 Description

Lists all applications that are currently deployed in the WebLogic domain.

In the event of an error, the command returns a WLSTException.

3.5.4.2 Syntax

```
listApplications()
```

3.5.4.3 Example

The following example lists all the applications currently deployed in mydomain.

```
wls:/mydomain/serverConfig> listApplications()
SamplesSearchWebApp
asyncServletEar
jspSimpleTagEar
ejb30
webservicesJwsSimpleEar
ejb20BeanMgedEar
xmlBeanEar
extServletAnnotationsEar
examplesWebApp
apache_xbean.jar
mainWebApp
jdbcRowSetsEar
```

3.5.5 loadApplication

Command Category: Deployment Commands

Use with WLST: Online and Offline

3.5.5.1 Description

Loads an application and deployment plan into memory. When used in online mode, you can connect only to the Administration Server; you cannot connect to a Managed Server.

The loadApplication command returns a WLSTPlan object that you can access to make changes to the deployment plan. For more information about the WLSTPlan object, see "WLSTPlan Object" in *Oracle WebLogic Scripting Tool*. In the event of an error, the command returns a WLSTException.

3.5.5.2 Syntax

```
loadApplication(appPath, [planPath], [createPlan])
```

Argument	Definition
appPath	Name of the top-level parent application directory, archive file, or root of the exploded archive directory containing the application to be loaded.
planPath	Optional. Name of the deployment plan file. The filename can be absolute or relative to the application directory. This argument defaults to the `plan/plan.xml` file in the application directory, if one exists.
createPlan	Optional. Boolean value specifying whether WLST should create a plan in the application directory if the specified plan does not exist. This argument defaults to `true`.

3.5.5.3 Example

The following example loads the `c:/myapps/myejb.jar` application using the plan file at `c:/myplans/myejb/plan.xml`.

```
wls:/offline> myPlan=loadApplication('c:/myapps/myejb.jar',
'c:/myplans/myejb/plan.xml')
Loading application from c:/myapps/myejb.jar and deployment plan from
c:/myplans/myejb/plan.xml ...
Successfully loaded the application.
```

The previous example stores the `WLSTPlan` object returned in the `myPlan` variable. You can then use `myPlan` variable to display information about the plan, such as the variables. For example:

```
wls:/offline> myPlan.showVariables()
MyEJB jndi.ejb
MyWAR app.foo
```

For more information about the `WLSTPlan` object, see "WLSTPlan Object" in *Oracle WebLogic Scripting Tool*.

3.5.6 redeploy

Command Category: Deployment Commands

Use with WLST: Online

3.5.6.1 Description

Reloads classes and redeploys a previously deployed application.

The `redeploy` command returns a `WLSTProgress` object that you can access to check the status of the command. For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*.

In the event of an error, the command returns a `WLSTException`.

For more information about redeploying applications, see "Overview of Common Deployment Scenarios" in *Deploying Applications to Oracle WebLogic Server*.

3.5.6.2 Syntax

redeploy(appName, [planPath], [options])

Argument	Definition
appName	Name of the application to be redeployed.

Argument	Definition
planPath	Optional. Name of the deployment plan file. The filename can be absolute or relative to the application directory. This argument defaults to the `plan/plan.xml` file in the application directory, if one exists.
options	Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see *options* argument description in Section 3.5.1, "deploy". In addition, the following deployment option can be specified for the `redeploy` command: ■ **appPath**—Name of the archive file or root of the exploded archive directory to be redeployed. ■ **deploymentPrincipalName**—String value specifying the principal for redeploying the file or archive during server starts. You can use this option to overwrite the current `<deployment-principal-name>` in the `config.xml` file.

3.5.6.3 Example

The following example redeploys `myApp` application using the `plan.xml` file located in the `c:/myapps` directory.

```
wls:/mydomain/serverConfig> progress=redeploy('myApp' 'c:/myapps/plan.xml')
Redeploying application 'myApp' ...
Redeployment of 'myApp' is successful
wls:/mydomain/serverConfig>
```

The previous example stores the `WLSTProgress` object returned in a user-defined variable, in this case, `progress`. You can then use the `progress` variable to access the state of the `redeploy` command. For example:

```
wls:/mydomain/serverConfig/Servers> progress.getState()
'completed'
wls:/mydomain/serverConfig/Servers>
```

For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*.

3.5.7 startApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.7.1 Description

Starts an application, making it available to users. The application must be fully configured and available in the WebLogic domain.

The `startApplication` command returns a `WLSTProgress` object that you can access to check the status of the command. For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*. In the event of an error, the command returns a `WLSTException`.

3.5.7.2 Syntax

```
startApplication(appName, [options])
```

Argument	Definition
appName	Name of the application to start, as specified in the plan.xml file.
options	Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see *options* argument description in Section 3.5.1, "deploy".

3.5.7.3 Example

The following example starts the BigApp application with the specified deployment options.

```
wls:/mydomain/serverConfig/Servers> progress=startApplication('BigApp',
stageMode='NOSTAGE', adminMode='false')
Starting the application...
Successfully started the application.
```

The previous example stores the WLSTProgress object returned in a user-defined variable, in this case, progress. You can then use the progress variable to access the state of the startApplication command. For example:

```
wls:/mydomain/serverConfig/Servers> progress.getState()
'completed'
wls:/mydomain/serverConfig/Servers>
```

For more information about the WLSTProgress object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*.

3.5.8 stopApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.8.1 Description

Stops an application, making it unavailable to users. The application must be fully configured and available in the WebLogic domain.

The stopApplication command returns a WLSTProgress object that you can access to check the status of the command. For more information about the WLSTProgress object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*.

In the event of an error, the command returns a WLSTException.

3.5.8.2 Syntax

stopApplication(appName, [options])

Argument	Definition
appName	Name of the application to stop, as specified in the plan.xml file.
options	Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see *options* argument description in Section 3.5.1, "deploy".

3.5.8.3 Example

The following example stops the BigApp application.

```
wls:/offline> progress=stopApplication('BigApp')
```

```
Stopping the application...
Successfully stopped the application.
```

The previous example stores the WLSTProgress object returned in a user-defined variable, in this case, progress. You can then use the progress variable to check whether stopApplication command is running. For example:

```
wls:/mydomain/serverConfig/Servers> progress.isRunning()
0
wls:/mydomain/serverConfig/Servers>
```

For more information about the WLSTProgress object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*.

3.5.9 undeploy

Command Category: Deployment Commands

Use with WLST: Online

3.5.9.1 Description

Undeploys an application from the specified servers.

The undeploy command returns a WLSTProgress object that you can access to check the status of the command. For more information about the WLSTProgress object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*. In the event of an error, the command returns a WLSTException.

For more information about deploying and undeploying applications, see "Overview of Common Deployment Scenarios" in *Deploying Applications to Oracle WebLogic Server*.

3.5.9.2 Syntax

undeploy(appName,[targets],[options])

Argument	Definition
appName	Deployment name for the deployed application.
targets	Optional. List of the target servers from which the application will be removed. If not specified, defaults to all current targets.
options	Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see *options* argument description in Section 3.5.1, "deploy".

3.5.9.3 Example

The following example removes the businessApp application from all target servers. WLST waits 60,000 ms for the process to complete.

```
wls:/mydomain/serverConfig> undeploy('businessApp', timeout=60000)
Undeploying application businessApp ...
<Jul 20, 2005 9:34:15 AM EDT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating undeploy operation for application, businessApp [archive: null],
to AdminServer .>
Completed the undeployment of Application with status
Current Status of your Deployment:
Deployment command type: undeploy
Deployment State      : completed
Deployment Message    : no message
```

```
wls:/mydomain/serverConfig>
```

3.5.10 updateApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.10.1 Description

Updates an application configuration using a new deployment plan. The application must be fully configured and available in the WebLogic domain.

The `updateApplication` command returns a `WLSTProgress` object that you can access to check the status of the command. For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*. In the event of an error, the command returns a `WLSTException`.

3.5.10.2 Syntax

```
updateApplication(appName, [planPath], [options])
```

Argument	Definition
appName	Name of the application, as specified in the current `plan.xml` file.
planPath	Optional. Name of the new deployment plan file. The filename can be absolute or relative to the application directory.
options	Optional. Comma-separated list of deployment options, specified as name-value pairs. For a list of valid deployment options, see *options* argument description in Section 3.5.1, "deploy".

3.5.10.3 Example

The following example updates the application configuration for `BigApp` using the `plan.xml` file located in `c:/myapps/BigApp/newPlan`.

```
wls:/offline> progress=updateApplication('BigApp',
'c:/myapps/BigApp/newPlan/plan.xml', stageMode='STAGE', adminMode='false')
Updating the application...
Successfully updated the application.
```

The previous example stores the `WLSTProgress` object returned in a user-defined variable, in this case, `progress`. You can then use the `progress` variable to access the state of the `updateApplication` command. For example:

```
wls:/mydomain/serverConfig/Servers> progress.getState()
'completed'
wls:/mydomain/serverConfig/Servers>
```

For more information about the `WLSTProgress` object, see "WLSTProgress Object" in *Oracle WebLogic Scripting Tool*.

3.6 Diagnostics Commands

Use the WLST diagnostics commands, listed in Table 3–6, to retrieve diagnostics data by executing queries against the WebLogic Diagnostics Framework (WLDF) data stores. For more information about WLDF, see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

Table 3–6   Diagnostic Command for WLST Configuration

This command...	Enables you to...	Use with WLST...
exportDiagnosticData	Execute a query against the specified log file.	Offline
exportDiagnosticDataFromServer	Executes a query on the server side and retrieves the exported WebLogic Diagnostic Framework (WLDF) data.	Online
getAvailableCapturedImages	Returns a list of the previously captured diagnostic images.	Online
saveDiagnosticImageCaptureFile	Downloads the specified diagnostic image capture.	Online
saveDiagnosticImageCaptureEntryFile	Downloads a specific entry from the diagnostic image capture.	Online

3.6.1 exportDiagnosticData

Command Category: Diagnostics Commands

Use with WLST: Offline

3.6.1.1 Description

Executes a query against the specified log file. The results are saved to an XML file.

For more information about the WebLogic Server Diagnostic Service, see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

In the event of an error, the command returns a `WLSTException`.

3.6.1.2 Syntax

```
exportDiagnosticData([options])
```

Argument	Definition
options	Optional. Comma-separated list of export diagnostic options, specified as name-value pairs. Valid options include: ■ **beginTimestamp**—Timestamp (inclusive) of the earliest record to be added to the result set. This option defaults to 0. ■ **endTimestamp**—Timestamp (exclusive) of the latest record to be added to the result set. This option defaults to `Long.MAX_VALUE`. ■ **exportFileName**—Name of the file to which the data is exported. This option defaults to `export.xml`. ■ **logicalName**—Logical name of the log file being read. Valid values include: `HarvestedDataArchive`, `EventsDataArchive`, `ServerLog`, `DomainLog`, `HTTPAccessLog`, `WebAppLog`, `ConnectorLog`, and `JMSMessageLog`. This option defaults to `ServerLog`. ■ **logName**—Base log filename containing the log data to be exported. This option defaults to `myserver.log`. ■ **logRotationDir**—Directory containing the rotated log files. This option defaults to `"."` (the same directory in which the log file is stored). ■ **query**—Expression specifying the filter condition for the data records to be included in the result set. This option defaults to `""` (empty string), which returns all data. For more information, see "WLDF Query Language" in *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*. ■ **storeDir**—Location of the diagnostic store for the server. This option defaults to `../data/store/diagnostics`.

3.6.1.3 Example

The following example executes a query against the `ServerLog` named `myserver.log` and stores the results in the file named `myExport.xml`.

```
wls:/offline/mydomain>exportDiagnosticData(logicalName='ServerLog',
logName='myserver.log', exportFileName='myExport.xml')
{'elfFields': '', 'logName': 'myserver.log', 'logRotationDir': '.',
'endTimestamp': 9223372036854775807L, 'exportFileName': 'export.xml',
'storeDir': '../data/store/diagnostics', 'logicalName': 'ServerLog',
'query': '', 'beginTimestamp': 0}

Exporting diagnostic data to export.xml
<Aug 2, 2005 6:58:21 PM EDT> <Info> <Store> <BEA-280050> <Persistent store
 "WLS_DIAGNOSTICS" opened: directory="c:\Oracle\Middleware
\wlserver_12.1\server\data\store\diagnostics"
 writePolicy="Disabled" blockSize=512 directIO=false driver="wlfileio2">

wls:/mydomain/serverRuntime>
```

3.6.2 exportDiagnosticDataFromServer

Command Category: Diagnostics Commands

Use with WLST: Online

3.6.2.1 Description

Executes a query on the server side and retrieves the exported WebLogic Diagnostic Framework (WLDF) data. The results are saved to an XML file.

For more information about the WebLogic Server Diagnostic Service, see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

In the event of an error, the command returns a `WLSTException`.

3.6.2.2 Syntax

```
exportDiagnosticDataFromServer([options])
```

Argument	Definition
options	Optional. Comma-separated list of export diagnostic options, specified as name-value pairs. Valid options include:
	■ **beginTimestamp**—Timestamp (inclusive) of the earliest record to be added to the result set. This option defaults to 0.
	■ **endTimestamp**—Timestamp (exclusive) of the latest record to be added to the result set. This option defaults to `Long.MAX_VALUE`.
	■ **exportFileName**—Name of the file to which the data is exported. This option defaults to `export.xml`.
	■ **logicalName**—Logical name of the log file being read. Valid values include: `HarvestedDataArchive`, `EventsDataArchive`, `ServerLog`, `DomainLog`, `HTTPAccessLog`, `WebAppLog`, `ConnectorLog`, and `JMSMessageLog`. This option defaults to `ServerLog`.
	■ **query**—Expression specifying the filter condition for the data records to be included in the result set. This option defaults to "" (empty string), which returns all data.

3.6.2.3 Example

The following example executes a query against the `HTTPAccessLog` and stores the results in the file named `myExport.xml`.

```
wls:/mydomain/serverRuntime>
exportDiagnosticDataFromServer(logicalName="HTTPAccessLog",
exportFileName="myExport.xml")
```

3.6.3 getAvailableCapturedImages

Command Category: Diagnostics Commands

Use with WLST: Online

3.6.3.1 Description

Returns, as an array of strings, a list of the previously captured diagnostic images that are stored in the image destination directory configured on the server. The default directory is *SERVER*`\logs\diagnostic_images`.

This command is useful for identifying a diagnostic image capture that you want to download, or for identifying a diagnostic image capture from which you want to download a specific entry.

For more information about the WebLogic Server Diagnostic Service, see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

In the event of an error, the command returns a `WLSTException`.

3.6.3.2 Syntax

```
getAvailableCapturedImages()
```

3.6.3.3 Example

The following example returns an array of strings named `images`, which contains a list of the diagnostic image capture files available in the image destination directory, and prints the entries contained in the diagnostic image named `diagnostic_image_ myserver_2009_06_15_14_58_36.zip`.

```
wls:/mydomain/serverRuntime>images=getAvailableCapturedImages()
Connecting to http://localhost:7001 with userid weblogic ...
wls:/mydomain/serverRuntime>print images [ 'diagnostic_image_myserver_2009_06_15_
14_58_36.zip' ]
```

3.6.4 saveDiagnosticImageCaptureFile

Command Category: Diagnostics Commands

Use with WLST: Online

3.6.4.1 Description

Downloads the specified diagnostic image capture from the server to which WLST is currently connected.

For more information about the WebLogic Server Diagnostic Service, see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

In the event of an error, the command returns a `WLSTException`.

3.6.4.2 Syntax

```
saveDiagnosticImageCaptureFile(imageName, [outputFile])
```

Argument	Definition
imageName	The name of the diagnostic image capture to download.
outputFile	Optional. Local path and file name in which the retrieved diagnostic image capture is to be stored. If not specified, this argument defaults to the value of `imageName` and the current working directory.

3.6.4.3 Example

The following example retrieves the list of the diagnostic image captures that are stored in the image destination directory on the server. It then shows two uses of the `saveDiagnosticImageCaptureFile` command. In the first use, the first diagnostic image capture in the list is downloaded to the local machine using the default output file name. In the second use, the first diagnostic image capture in the list is downloaded to the local machine in the file `mylocalimg.zip`.

```
wls:/mydomain/serverRuntime>images=getAvailableCapturedImages()
Connecting to http://localhost:7001 with userid weblogic ...
wls:/mydomain/serverConfig> saveDiagnosticImageCaptureFile(images[0])
Retrieving diagnostic_image_myserver_2009_06_25_12_12_50.zip to local
path diagnostic_image_myserver_2009_06_25_12_12_50.zip
Connecting to http://localhost:7001 with userid weblogic ...
wls:/mydomain/serverConfig> saveDiagnosticImageCaptureFile(images[0],
'mylocalimg.zip')
Retrieving diagnostic_image_myserver_2009_06_25_12_12_50.zip to local
path mylocalimg.zip
Connecting to http://localhost:7001 with userid weblogic ...
```

3.6.5 saveDiagnosticImageCaptureEntryFile

Command Category: Diagnostics Commands

Use with WLST: Online

3.6.5.1 Description

Downloads a specific entry from the diagnostic image capture that is located on the server to which WLST is currently connected.

For more information about the WebLogic Server Diagnostic Service, see *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

In the event of an error, the command returns a `WLSTException`.

3.6.5.2 Syntax

saveDiagnosticImageCaptureEntryFile(imageName, imageEntryName, [outputFile])

Argument	Definition
imageName	Name of the diagnostic image capture containing the desired entry.

imageEntryName	Name of the specific entry to be retrieved from the diagnostic image capture. This can be one of the following:
	
	`image.summary`
	`JTA.img`
	`JRockitFlightRecorder.jfr`
	`FlightRecording.jfr`
	`WatchSource.img`
	`configuration.img`
	`WORK_MANAGER.img`
	`JNDI_IMAGE_SOURCE.img`
	`APPLICATION.img`
	`InstrumentationImageSource.img`
	`SAF.img`
	`Logging.img`
	`PERSISTENT_STORE.img`
	`JDBC.img`
	`PathService.img`
	`JMS.img`
	`Deployment.img`
	`JVM.img`
	`CONNECTOR.img`

outputFile	Optional. Local path and file name in which the entry retrieved from the diagnostic image capture is to be stored. If not specified, this argument defaults to the value of `imageEntryName` and the current working directory.

3.6.5.3 Example

The following example gets the list of diagnostic image captures, then uses the `saveDiagnosticImageCaptureEntryFile` twice. In the first use, this example retrieves the image summary to the local machine using the default output file name. In the second use, it retrieves the image summary to the local machine in the file `myimage.summary`.

```
wls:/mydomain/serverRuntime>images=getAvailableCapturedImages()
Connecting to http://localhost:7001 with userid weblogic ...
wls:/mydomain/serverConfig> saveDiagnosticImageCaptureEntryFile(images[0],
'image.summary')
```

```
Retrieving entry image.summary from diagnostic_image_myserver_2009_06_25_12_12_
50.zip to local path image.summary
Connecting to http://localhost:7001 with userid weblogic ...
wls:/mydomain/serverConfig> saveDiagnosticImageCaptureEntryFile(images[0],
'image.summary', 'myimage.summary')
Retrieving entry image.summary from diagnostic_image_myserver_2009_06_25_12_12_
50.zip to local path myimage.summary
Connecting to http://localhost:7001 with userid weblogic ...
```

3.7 Editing Commands

Use the WLST editing commands, listed in Table 3–7, to interrogate and edit configuration beans.

> **Note:** To edit configuration beans, you must be connected to an Administration Server, and you must navigate to the edit tree and start an edit session, as described in Section 3.11.5, "edit" and Section 3.7.17, "startEdit", respectively.
>
> If you connect to a Managed Server, WLST functionality is limited to browsing the configuration bean hierarchy. While you cannot use WLST to change the values of MBeans on Managed Servers, it is possible to use the Management APIs to do so. Oracle recommends that you change only the values of configuration MBeans on the Administration Server. Changing the values of MBeans on Managed Servers can lead to an inconsistent domain configuration.
>
> For more information about editing configuration beans, see "Using WLST Online to Update an Existing Domain" in *Oracle WebLogic Scripting Tool*.

Table 3–7 Editing Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
activate	Activate changes saved during the current editing session but not yet deployed.	Online or Offline
assign	Assign resources to one or more destinations.	Offline
cancelEdit	Cancel an edit session, release the edit lock, and discard all unsaved changes. This operation can be called by any user with administrator privileges, even if the user did not start the edit session.	Online
create	Create a configuration bean of the specified type for the current bean.	Online or Offline
delete	Delete an instance of a configuration for the current configuration bean.	Online or Offline
encrypt	Encrypt the specified string.	Online
get	Return the value of the specified attribute.	Online or Offline
getActivationTask	Return the latest ActivationTask MBean on which a user can get status.	Online
invoke	Invokes a management operation on the current configuration bean.	Online

Table 3–7 (Cont.) Editing Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
isRestartRequired	Determine whether a server restart is required.	Online
loadDB	Load SQL files into a database.	Offline
loadProperties	Load property values from a file.	Online or Offline
save	Save the edits that have been made but have not yet been saved.	Online
set	Set the specified attribute value for the current configuration bean.	Online or Offline
setOption	Set options related to a WebLogic domain creation or update.	Offline
showChanges	Show the changes made to the configuration by the current user during the current edit session.	Online
startEdit	Starts a configuration edit session on behalf of the currently connected user.	Online
stopEdit	Stop the current edit session, release the edit lock, and discard unsaved changes.	Online
unassign	Unassign applications or resources from one or more destinations.	Offline
undo	Revert all unsaved or unactivated edits.	Online
validate	Validate the changes that have been made but have not yet been saved.	Online

3.7.1 activate

Command Category: Editing Commands

Use with WLST: Online

3.7.1.1 Description

Activates changes saved during the current editing session but not yet deployed. This command prints a message if a server restart is required for the changes that are being activated.

The `activate` command returns the latest `ActivationTask` MBean which reflects the state of changes that a user is currently making or has made recently. You can then invoke methods to get information about the latest Configuration Manager activate task in progress or just completed. In the event of an error, the command returns a `WLSTException`.

3.7.1.2 Syntax

activate([timeout], [block])

Argument	Definition
timeout	Optional. Time (in milliseconds) that WLST waits for the activation of configuration changes to complete before canceling the operation. A value of -1 indicates that the operation will not time out. This argument defaults to 300,000 ms (or 5 minutes).

Argument	Definition
block	Optional. Boolean value specifying whether WLST should block user interaction until the command completes. This argument defaults to `false`, indicating that user interaction is not blocked. In this case, WLST returns control to the user after issuing the command and assigns the task MBean associated with the current task to a variable that you can use to check its status.If you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*, *block* is always set to `true`.

3.7.1.3 Example

The following example activates the changes made during the current edit session that have been saved to disk, but that have not yet been activated. WLST waits for 100,000 ms for the activation to complete, and 200,000 ms before the activation is stopped.

```
wls:/mydomain/edit !> activate(200000, block='true')
Activating all your changes, this may take a while ...
The edit lock associated with this edit session is released once the activation is
completed.
Action completed.
wls:/mydomain/edit>
```

3.7.2 assign

Command Category: Editing Commands

Use with WLST: Offline

3.7.2.1 Description

Assigns resources to one or more destinations.

In the event of an error, the command returns a `WLSTException`.

3.7.2.2 Syntax

assign(sourceType, sourceName, destinationType, destinationName)

Argument	Definition
sourceType	Type of configuration bean to be assigned. This value can be set to one of the following values: ■ `AppDeployment` ■ `Library` ■ `securityType` (such as `User`) ■ `Server` ■ `service` (such as `JDBCSystemResource`) ■ *service*`.SubDeployment`, where *service* specifies the service type of the `SubDeployment` (such as `JMSSystemResource.SubDeployment`); you can also specify nested subdeployments (such as `AppDeployment.SubDeployment.SubDeployment`) Guidelines for setting this value are provided below.

Argument	Definition
sourceName	Name of the resource to be assigned. Multiple names can be specified, separated by commas, or you can use the wildcard (*) character to specify all resources of the specified type.
	Specify subdeployments using the following format: *service.subDeployment*, where *service* specifies the parent service and *subDeployment* specifies the name of the subdeployment. For example, `myJMSResource.myQueueSubDeployment`. You can also specify nested subdeployments, such as `MedRecEAR.MedRecAppScopedJMS.MedRecJMSServer`.
	Note: A given subdeployment name cannot contain a dot (.), as the `assign` command will interpret it as a nested subdeployment.
destinationType	Type of destination. Guidelines for setting this value are provided below.
destinationName	Name of the destination. Multiple names can be specified, separated by commas.

Use the following guidelines for setting the `sourceType` and `destinationType`:

- When assigning **application deployments**, set the values as follows:

  - *sourceType*: `AppDeployment`

  - *destinationType*: `Target`

- When assigning **libraries**, set the values as follows:

  - *sourceType*: `Library`

  - *destinationType*: `Target`

- When assigning **services**, set the values as follows:

  - *sourceType*: Name of the specific server, such as `JDBCSystemResource`

  - *destinationType*: `Target`

- When assigning **servers** to **clusters**, set the values as follows:

  - *sourceType*: `Server`

  - *destinationType*: `Cluster`

- When assigning **subdeployments**, set the values as follows:

  - *sourceType*: *service*.`SubDeployment`, where *service* specifies the parent of the `SubDeployment`, such as `JMSSystemResource.SubDeployment`; you can also specify nested subdeployments (such as `AppDeployment.SubDeployment.SubDeployment`)

  - *destinationType*: `Target`

- When assigning **security types**, set the values as follows:

  - *sourceType*: Name of the security type, such as `User`

  - *destinationType*: Name of the destination security type, such as `Group`

3.7.2.3 Example

The following examples:

- Assign the servers `myServer` and `myServer2` to the cluster `myCluster`.

```
wls:/offline/mydomain> assign("Server", "myServer,myServer2", "Cluster",
"myCluster")
```

■ Assign all servers to the cluster `myCluster`.

```
wls:/offline/mydomain> assign("Server", "*", "Cluster", "myCluster")
```

■ Assign the application deployment `myAppDeployment` to the target server `newServer`.

```
wls:/offline/mydomain> assign("AppDeployment", "myAppDeployment", "Target",
"newServer")
```

■ Assign the user `newUser` to the group `Monitors`.

```
wls:/offline/mydomain> assign("User", "newUser", "Group", "Monitors")
```

■ Assign the SubDeployment `myQueueSubDeployment`, which is a child of the JMS resource `myJMSResource`, to the target server `newServer`.

```
wls:/offline/mydomain> assign('JMSSystemResource.SubDeployment',
'myJMSResource.myQueueSubDeployment', 'Target', 'newServer')
```

■ Assign the nested SubDeployment `MedRecAppScopedJMS.MedRecJMSServer`, which is a child of the AppDeployment `AppDeployment`, to the target server `AdminServer`.

```
wls:/offline/mydomain>assign('AppDeployment.SubDeployment.SubDeployment
','MedRecEAR.MedRecAppScopedJMS.MedRecJMSServer','Target','AdminServer')
```

3.7.3 cancelEdit

Command Category: Editing Commands

Use with WLST: Online

3.7.3.1 Description

Cancels an edit session, releases the edit lock, and discards all unsaved changes.

The user issuing this command does not have to be the current editor; this allows an administrator to cancel an edit session, if necessary, to enable other users to start an edit session.

In the event of an error, the command returns a `WLSTException`.

3.7.3.2 Syntax

cancelEdit([defaultAnswer])

Argument	Definition
defaultAnswer	Optional. Default response, if you would prefer not to be prompted at the command line. Valid values are y and n. This argument defaults to null, and WLST prompts you for a response.

3.7.3.3 Example

The following example cancels the current editing session. WLST prompts for verification before canceling.

```
wls:/mydomain/edit !> cancelEdit()
Sure you would like to cancel the edit session? (y/n)y
```

```
Edit session is cancelled successfully
wls:/mydomain/edit>
```

3.7.4 create

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.4.1 Description

Creates a configuration bean of the specified type for the current bean.

The `create` command returns a stub for the newly created configuration bean. In the event of an error, the command returns a `WLSTException`.

> **Note:** Child types must be created under an instance of their parent type. You can only create configuration beans that are children of the current Configuration Management Object (`cmo`) type. For more information about the `cmo` variable, see "Changing the Current Management Object" in *Oracle WebLogic Scripting Tool*.

Please note the following when using the `create` command with **WLST online**:

- You must be connected to an Administration Server. You cannot use the `create` command for runtime MBeans or when WLST is connected to a Managed Server instance.

- You must navigate to the edit configuration MBean hierarchy using the `edit` command before issuing this command. See Section 3.11.5, "edit".

- You can use the create command to create a WebLogic Server configuration MBean that is a child of the current MBean type.

Please note the following when using the `create` command with **WLST offline**:

- When using WLST offline, the following characters are not valid in object names: period (`.`), forward slash (`/`), or backward slash (`\`).

For more information about:

- Creating MBeans, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities with JMX*.

- Examples of creating specific types of MBean resources, for example, a JMS or JDBC system resource, refer to the WLST sample scripts installed with your product, as described in "WLST Sample Scripts" in *Oracle WebLogic Scripting Tool*.

- MBeans, their child types, attributes, and operations, see *Oracle WebLogic Server MBean Reference*.

3.7.4.2 Syntax

create(name, childMBeanType, [baseProviderType])

Argument	Definition
name	Name of the configuration bean that you are creating.

Argument	Definition
childMBeanType	Type of configuration bean that you are creating. You can create instances of any type defined in the config.xml file except custom security types. For more information about valid configuration beans, see *Oracle WebLogic Server MBean Reference*.
baseProviderType	When creating a security provider, specifies the base security provider type, for example, AuthenticationProvider. This argument defaults to None.

3.7.4.3 Example

The following example creates a child configuration bean of type Server named newServer for the current configuration bean, storing the stub as server1:

```
wls:/mydomain/edit !> server1=create('newServer','Server')
Server with name 'newServer' has been created successfully.
wls:/mydomain/edit !> server1.getName()
'newServer'
wls:/mydomain/edit !>
```

The following example creates an authentication provider security provider called myProvider:

```
wls:/mydomain/edit !> cd('SecurityConfiguration/mydomain/Realms/myrealm')
wls:/mydomain/edit !>
create('myProvider','weblogic.security.providers.authentication.SQLAuthenticator'
,'AuthenticationProvider')
wls:/mydomain/edit ! cd('AuthenticationProviders/myProvider')
wls:/mydomain/edit ! set('ControlFlag', 'REQUIRED')
```

The following example creates a machine named highsec_nm and sets attributes for the associated Node Manager.

```
wls:/mydomain/edit !> create('highsec_nm', 'Machine')
wls:/mydomain/edit !> cd('Machine/highsec_nm/NodeManager/highsec_nm')
wls:/mydomain/edit !> set('DebugEnabled', 'true')
wls:/mydomain/edit !> set('ListenAddress', 'innes')
wls:/mydomain/edit !> set('NMType', 'SSL')
wls:/mydomain/edit !> set('ShellCommand', '')
```

3.7.5 delete

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.5.1 Description

Deletes an instance of a configuration bean of the specified type for the current configuration bean.

In the event of an error, the command returns a WLSTException.

> **Note:** You can only delete configuration beans that are children of current Configuration Management Object (cmo) type. For more information about the cmo variable, see "Changing the Current Management Object" in *Oracle WebLogic Scripting Tool*.

3.7.5.2 Syntax

```
delete(name, childMBeanType)
```

Argument	Definition
name	Name of the child configuration bean to delete.
childMBeanType	Type of the configuration bean to be deleted. You can delete instances of any type defined in the config.xml file. For more information about valid configuration beans, see *Oracle WebLogic Server MBean Reference*.

3.7.5.3 Example

The following example deletes the configuration bean of type Server named newServer:

```
wls:/mydomain/edit !> delete('newServer','Server')
Server with name 'newServer' has been deleted successfully.
wls:/mydomain/edit !>
```

3.7.6 encrypt

Command Category: Editing Commands

Use with WLST: Online

3.7.6.1 Description

Encrypts the specified string. You can then use the encrypted string in your configuration file or as an argument to a command.

You must invoke this command once for each WebLogic domain in which you want to use the encrypted string. The string can be used only in the WebLogic domain for which it was originally encrypted.

In the event of an error, the command returns a WLSTException.

3.7.6.2 Syntax

```
encrypt(obj, [domainDir])
```

Argument	Definition
obj	String that you want to encrypt.
domainDir	Optional. Absolute path name of a WebLogic domain directory. The encrypted string can be used only by the WebLogic domain that is contained within the specified directory.
	If you do not specify this argument, the command encrypts the string for use in the WebLogic domain to which WLST is currently connected.

3.7.6.3 Example

The following example encrypts the specified string using the security/SerializedSystemIni.dat file in the specified WebLogic domain directory.

```
wls:/mydomain/serverConfig>
es=encrypt('myPassword','c:/Oracle/Middleware/domains/mydomain')
```

3.7.7 get

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.7.1 Description

Returns the value of the specified attribute. For more information about the MBean attributes that can be viewed, see *Oracle WebLogic Server MBean Reference*. In the event of an error, the command returns a `WLSTException`.

> **Note:** You can list all attributes and their current values by entering `ls('a')`. For more information, see Section 3.8.12, "ls".

Alternatively, you can use the `cmo` variable to perform any get method on the current configuration bean. For example:

```
cmo.getListenPort()
```

For more information about the `cmo` variable, see "Changing the Current Management Object" in *Oracle WebLogic Scripting Tool*.

3.7.7.2 Syntax

```
get(attrName)
```

Argument	Definition
attrName	Name of the attribute to be displayed. You can specify the full pathname of the attribute. If no pathname is specified, the attribute is displayed for the current configuration object.

3.7.7.3 Example

The following example returns the value of the `AdministrationPort` for the current configuration bean.

```
wls:/mydomain/serverConfig> get('AdministrationPort')
9002
```

Alternatively, you can use the `cmo` variable:

```
cmo.getAdministrationPort()
```

3.7.8 getActivationTask

Command Category: Editing Commands

Use with WLST: Online

3.7.8.1 Description

Return the latest `ActivationTask` MBean on which a user can get status. The `ActivationTask` MBean reflects the state of changes that a user has made recently in WLST. You can then invoke methods to get information about the latest Configuration Manager activate task in progress or just completed. In the event of an error, the command returns a `WLSTException`.

> **Note:** If you have activated changes outside of WLST, use the
> ConfigurationManagerMBean getActivationTasks() method to get
> access to Activation Tasks created in other tools.

3.7.8.2 Syntax

```
getActivationTask()
```

3.7.8.3 Example

The following example returns the latest `ActivationTask` MBean on which a user
can get status and stores it within the task variable.

```
wls:/mydomain/edit> task=getActivationTask()
wls:/mydomain/edit> if task!=None:
...    task.getState()
...
4
```

3.7.9 invoke

Command Category: Editing Commands

Use with WLST: Online

3.7.9.1 Description

Invokes a management operation on the current configuration bean. Typically, you use
this command to invoke operations other than the `get` and `set` operations that most
WebLogic Server configuration beans provide. The class objects are loaded through the
same class loader that is used for loading the configuration bean on which the action is
invoked.

You cannot use the `invoke` command when WLST is connected to a Managed Server
instance.

If successful, the `invoke` command returns the object that is returned by the operation
invoked. In the event of an error, the command returns a `WLSTException`.

3.7.9.2 Syntax

```
invoke(methodName, parameters, signatures)
```

Argument	Definition
methodName	Name of the method to be invoked.
parameters	An array of parameters to be passed to the method call.
signatures	An array containing the signature of the action.

3.7.9.3 Example

The following example invokes the `lookupServer` method on the current
configuration bean.

```
wls:/mydomain/config> objs =
jarray.array([java.lang.String("oamserver")],java.lang.Object)
wls:/mydomain/edit> strs = jarray.array(["java.lang.String"],java.lang.String)
wls:/mydomain/edit> invoke('lookupServer',objs,strs)
true
```

```
wls:/mydomain/edit>
```

3.7.10 isRestartRequired

Command Category: Editing Commands

Use with WLST: Online

3.7.10.1 Description

Determines whether a server restart is required.

If you invoke this command while an edit session is in progress, the response is based on the edits that are currently in progress. If you specify the name of an attribute, WLST indicates whether a server restart is required for that attribute only.

In the event of an error, the command returns a WLSTException.

3.7.10.2 Syntax

```
isRestartRequired([attributeName])
```

Argument	Definition
attributeName	Optional. Name of a specific attribute for which you want to check if a server restart is required.

3.7.10.3 Example

The following example specifies whether a server restart is required for all changes made during the current WLST session.

```
wls:/mydomain/edit !> isRestartRequired()
Server re-start is REQUIRED for the set of changes in progress.

The following attribute(s) have been changed on MBeans that require server
re-start.
MBean Changed : mydomain:Name=mydomain,Type=Domain
Attributes changed : AutoConfigurationSaveEnabled
```

The following example specifies whether a server restart is required if you edit the ConsoleEnabled attribute.

```
wls:/mydomain/edit !> isRestartRequired("ConsoleEnabled")
Server re-start is REQUIRED if you change the attribute ConsoleEnabled
wls:/mydomain/edit !>
```

3.7.11 loadDB

Command Category: Editing Commands

Use with WLST: Offline

3.7.11.1 Description

Loads SQL files into a database.

The loadDB command loads the SQL files from a template file. This command can only be issued after a domain template or extension template has been loaded into memory (see Section 3.3.8, "readDomain" and Section 3.3.9, "readTemplate").

Before executing this command, ensure that the following conditions are true:

- The appropriate database is running.

- SQL files exist for the specified database and version.

  To verify that the appropriate SQL files exist, open the domain template and locate the relevant SQL file list, jdbc.index, in the _jdbc_ directory. For example, for Oracle 9*i*, the SQL file list is located at _jdbc_\Oracle\9i\jdbc.index.

The command fails if the above conditions are not met.

In the event of an error, the command returns a WLSTException.

3.7.11.2 Syntax

loadDB(dbVersion, datasourceName, dbCategory)

Argument	Definition
dbVersion	Version of the database for which the SQL files are intended to be used.
datasourceName	Name of the JDBC data source to be used to load SQL files.
dbCategory	Optional. Database category associated with the specified data source.
	For more information about the jdbc.index file and database categories, see "Files Typically Included in a Template" in the *Oracle WebLogic Server Domain Template Reference*.

3.7.11.3 Example

The following example loads SQL files related to Drop/Create P13N Database Objects intended for version 5.1 of the database, using the p13nDataSource JDBC data source.

```
wls:/offline/mydomain> loadDB('5.1', 'p13nDataSource', 'Drop/Create P13N Database
Objects')
```

3.7.12 loadProperties

Command Category: Editing Commands

Use with WLST: Online and Offline

3.7.12.1 Description

Loads property values from a file and makes them available in the WLST session.

This command cannot be used when you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*.

In the event of an error, the command returns a WLSTException.

3.7.12.2 Syntax

loadProperties(fileName)

Argument	Definition
fileName	Properties file pathname.

3.7.12.3 Example

This example gets and sets the properties file values.

```
wls:/mydomain/serverConfig> loadProperties('c:/temp/myLoad.properties')
```

3.7.13 save

Command Category: Editing Commands

Use with WLST: Online

3.7.13.1 Description

Saves the edits that have been made but have not yet been saved. This command is only valid when an edit session is in progress. For information about starting an edit session, see Section 3.7.17, "startEdit".

In the event of an error, the command returns a WLSTException.

3.7.13.2 Syntax

```
save()
```

3.7.13.3 Example

The following example saves the edits that have not yet been saved to disk.

```
wls:/mydomain/edit !> save()
Saving all your changes ...
Saved all your changes successfully.
wls:/mydomain/edit !>
```

3.7.14 set

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.14.1 Description

Sets the value of a specified attribute in the current management object. When using WLST offline, this command writes the attribute value to the domain configuration files. When using WLST online, this command sets the value of an MBean attribute. Online changes are written to the domain configuration file when you activate your edits.

In the event of an error, the command returns a WLSTException.

For information about setting encrypted attributes (all encrypted attributes have names that end with Encrypted), see "Writing and Reading Encrypted Configuration Values" in *Oracle WebLogic Scripting Tool*.

Note the following when using **WLST online**:

- You must be in an edit session to use this command. See Section 3.7.17, "startEdit".

- You cannot use this command when WLST is connected to a Managed Server.

- As an alternative to this command, you can use the cmo variable with the following syntax:

  ```
  cmo.setattrName(value)
  ```

  For example, instead of using set('ListenPort', 7011), you can use:

  ```
  cmo.setListenPort(7011)
  ```

For more information about the cmo variable, see "Changing the Current Management Object" in *Oracle WebLogic Scripting Tool*.

3.7.14.2 Syntax

```
set(attrName, value)
```

Argument	Definition
attrName	Name of the attribute to be set.
value	Value of the attribute to be set.
	Note: This value should *not* be enclosed in single or double quotes. See the examples.

3.7.14.3 Example

The following example sets the ArchiveConfigurationCount attribute of DomainMBean to 10:

```
wls:/mydomain/serverConfig> set('ArchiveConfigurationCount', 10)
```

The following example sets the long value of the T1TimerInterval attribute of a custom Mbean to 123:

```
wls:/mydomain/serverConfig> set('T1TimerInterval', Long(123))
```

The following example sets the boolean value of the MyBooleanAttribute attribute of a custom Mbean to true:

```
wls:/mydomain/serverConfig> set('MyBooleanAttribute', Boolean(true))
```

3.7.15 setOption

Command Category: Editing Commands

Use with WLST: Offline

3.7.15.1 Description

Sets options related to a WebLogic domain creation or update. In the event of an error, the command returns a WLSTException.

3.7.15.2 Syntax

```
setOption(optionName, optionValue)
```

Argument	Definition
optionName	Name of the option to set.
	Available options for **domain creation** include:
	■ `CreateStartMenu`—Boolean value specifying whether to create a Start Menu shortcut on a Windows platform. This option defaults to `true`.
	Note: If a user with Administrator privileges installed the software and chose to create the Start menu entries in the All Users folder, only users with Administrator privileges can create Start menu entries in the same folder when creating a WebLogic domain using the Configuration Wizard or WLST. That is, if a user without Administrator privileges uses the Configuration Wizard or WLST from this installation to create domains, Start menu shortcuts to the domains are not created. In this case, the users can manually create shortcuts in their local Start menu folder, if desired.
	■ `DomainName`—Name of the WebLogic domain. By default, the name of the WebLogic domain is derived from the name of the domain directory. For example, for a WebLogic domain saved to `c:/Oracle/Middleware/user_projects/domains/myMedrec`, the domain name is `myMedrec`. By setting `DomainName`, the name of the created domain will be independent of the domain directory name.
	■ `JavaHome`—Home directory for the JVM to be used when starting the server. The default for this option depends on the platform on which you install WebLogic Server.
	■ `OverwriteDomain`—Boolean value specifying whether to allow an existing WebLogic domain to be overwritten. This option defaults to `false`.
	■ `ServerStartMode`—Mode to use when starting the server for the newly created WebLogic domain. This value can be `dev` (development) or `prod` (production). This option defaults to `dev`.
	Available options for **domain updates** include:
	■ `AllowCasualUpdate`—Boolean value specifying whether to allow a WebLogic domain to be updated without adding an extension template. This option defaults to `true`.
	■ `ReplaceDuplicates`—Boolean value specifying whether to keep original configuration elements in the WebLogic domain or replace the elements with corresponding ones from an extension template when there is a conflict. This option defaults to `true`.
	Available options for both **domain creation** and **domain updates** include:
	■ `AppDir`—Application directory to be used when a separate directory is desired for applications, as specified by the template. This option defaults to *WL_HOME*/user_projects/applications/*domainname*, where *WL_HOME* specifies the WebLogic Server home directory and *domainname* specifies the name of the WebLogic domain.
	■ `AutoAdjustSubDeploymentTarget`—Boolean value specifying whether WLST automatically adjusts targets for the subdeployments of AppDeployments. This option defaults to `true`. To deactivate this feature, set the option to `false` and explicitly set the targeting for AppDeployment subdeployments before writing or updating the WebLogic domain or domain template.
	■ `AutoDeploy`—Boolean value specifying whether to activate auto deployment when a cluster or multiple Managed Servers are created. This option defaults to `true`. To deactivate this feature, set the option to `false` on the first line of your script.
optionValue	Value for the option.
	Note: Boolean values can be specified as a String (`true`, `false`) or integer (`0`, `1`).

3.7.15.3 Example

The following example sets the CreateStartMenu option to false:

```
wls:/offline> setOption('CreateStartMenu', 'false')
```

3.7.16 showChanges

Command Category: Editing Commands

Use with WLST: Online

3.7.16.1 Description

Shows the changes made to the configuration by the current user during the current edit session. In the event of an error, the command returns a WLSTException.

3.7.16.2 Syntax

showChanges([onlyInMemory])

Argument	Definition
onlyInMemory	Optional. Boolean value specifying whether to display only the changes that have not yet been saved. This argument defaults to false, indicating that all changes that have been made from the start of the session are displayed.

3.7.16.3 Example

The following example shows all of the changes made by the current user to the configuration since the start of the current edit session.

```
wls:/mydomain/edit !> showChanges()
Changes that are in memory and saved to disc but not yet activated are:

MBean Changed           : com.bea:Name=basicWLSDomain,Type=Domain
Operation Invoked       : add
Attribute Modified       : Machines
Attributes Old Value     : null
Attributes New Value     : Mach1
Server Restart Required : false

MBean Changed           : com.bea:Name=basicWLSDomain,Type=Domain
Operation Invoked       : add
Attribute Modified       : Servers
Attributes Old Value     : null
Attributes New Value     : myserver
Server Restart Required : false
```

3.7.17 startEdit

Command Category: Editing Commands

Use with WLST: Online

3.7.17.1 Description

Starts a configuration edit session on behalf of the currently connected user. You must navigate to the edit configuration MBean hierarchy using the edit command before issuing this command. For more information, see Section 3.11.5, "edit".

This command must be called prior to invoking any command to modify the WebLogic domain configuration.

In the event of an error, the command returns a WLSTException.

Note: WLST automatically starts an edit session if it detects that there is an edit session that is already in progress by the same user, which may have been started via the Administration Console or another WLST session.

3.7.17.2 Syntax

startEdit([waitTimeInMillis], [timeoutInMillis], [exclusive])

Argument	Definition
waitTimeInMillis	Optional. Time (in milliseconds) that WLST waits until it gets a lock, in the event that another user has a lock. This argument defaults to 0 ms.
timeOutInMillis	Optional. Timeout (in milliseconds) that WLST waits to release the edit lock. This argument defaults to -1 ms, indicating that this edit session never expires.
exclusive	Optional. Specifies whether the edit session should be an exclusive session. If set to true, if the same owner enters the startEdit command, WLST waits until the current edit session lock is released before starting the new edit session. The exclusive lock times out according to the time specified in *timeoutInMillis*. This argument defaults to false.

3.7.17.3 Example

The following example saves the edits that have not yet been saved to disk.

```
wls:/mydomain/edit> startEdit(60000, 120000)
Starting an edit session ...
Started edit session, please be sure to save and activate your changes once you
are done.
wls:/mydomain/edit !>
```

3.7.18 stopEdit

Command Category: Editing Commands

Use with WLST: Online

3.7.18.1 Description

Stops the current edit session, releases the edit lock, and discards unsaved changes.

In the event of an error, the command returns a WLSTException.

3.7.18.2 Syntax

stopEdit([defaultAnswer])

Argument	Definition
defaultAnswer	Optional. Default response, if you would prefer not to be prompted at the command line. Valid values are y and n. This argument defaults to null, and WLST prompts you for a response.

3.7.18.3 Example

The following example stops the current editing session. WLST prompts for verification before canceling.

```
wls:/mydomain/edit !> stopEdit()
Sure you would like to stop your edit session? (y/n)
y
Edit session has been stopped successfully.
wls:/mydomain/edit>
```

3.7.19 unassign

Command Category: Editing Commands

Use with WLST: Offline

3.7.19.1 Description

Unassign applications or resources from one or more destinations.

In the event of an error, the command returns a `WLSTException`.

3.7.19.2 Syntax

`unassign(sourceType, sourceName, destinationType, destinationName)`

Argument	Definition
sourceType	Type of configuration bean to be unassigned. This value can be set to one of the following values: ■ `AppDeployment` ■ `Library` ■ *securityType* (such as `User`) ■ `Server` ■ *service* (such as `JDBCSystemResource`) ■ *service*.`SubDeployment`, where *service* specifies the service type of the `SubDeployment` (such as `JMSSystemResource.SubDeployment`); you can also specify nested subdeployments (such as `AppDeployment.SubDeployment.SubDeployment`)
sourceName	Name of the application or resource to be unassigned. Multiple names can be specified, separated by commas, or you can use the wildcard (*) character to specify all resources of the specified type. Specify subdeployments using the following format: *service.subDeployment*, where *service* specifies the parent service and *subDeployment* specifies the name of the subdeployment. For example, `myJMSResource.myQueueSubDeployment.` You can also specify nested subdeployments, such as `MedRecEAR.MedRecAppScopedJMS.MedRecJMSServer.`
destinationType	Type of destination. Guidelines for setting this value are provided below.
destinationName	Name of the destination. Multiple names can be specified, separated by commas.

Use the following guidelines for setting the `sourceType` and `destinationType`:

■ When unassigning **application deployments**, set the values as follows:

- – *sourceType*: AppDeployment

- – *destinationType*: Target

- When unassigning **libraries**, set the values as follows:

  - – *sourceType*: Library

  - – *destinationType*: Target

- When unassigning **security types**, set the values as follows:

  - – *sourceType*: Name of the security type, such as User

  - – *destinationType*: Name of the destination security type, such as Group

- When unassigning **servers** from **clusters**, set the values as follows:

  - – *sourceType*: Server

  - – *destinationType*: Cluster

- When unassigning **services**, set the values as follows:

  - – *sourceType*: Name of the specific server, such as JDBCSystemResource

  - – *destinationType*: Target

- When unassigning **subdeployments**, set the values as follows:

  - – *sourceType*: *service*.SubDeployment, where *service* specifies the parent of the SubDeployment, such as JMSSystemResource.SubDeployment; you can also specify nested subdeployments (such as AppDeployment.SubDeployment.SubDeployment)

  - – *destinationType*: Target

3.7.19.3 Example

The following examples:

- Unassign the servers myServer and myServer2 from the cluster myCluster.

  ```
  wls:/offline/medrec> unassign("Server", "myServer,myServer2", "Cluster",
  "myCluster")
  ```

- Unassign all servers from the cluster myCluster.

  ```
  wls:/offline/mydomain> unassign("Server", "*", "Cluster", "myCluster")
  ```

- Unassign the user newUser from the group Monitors.

  ```
  wls:/offline/medrec> unassign("User", "newUser", "Group", "Monitors")
  ```

- Unassign the application deployment myAppDeployment from the target server newServer.

  ```
  wls:/offline/mydomain> unassign("AppDeployment", "myAppDeployment", "Target",
  "newServer")
  ```

- Unassign the nested SubDeployment MedRecAppScopedJMS.MedRecJMSServer, which is a child of the AppDeployment AppDeployment, from the target server AdminServer.

  ```
  wls:/offline/mydomain> assign('AppDeployment.SubDeployment.SubDeployment',
  'MedRecEAR.MedRecAppScopedJMS.MedRecJMSServer', 'Target','AdminServer')
  ```

3.7.20 undo

Command Category: Editing Commands

Use with WLST: Online

3.7.20.1 Description

Reverts all unsaved or unactivated edits.

You specify whether to revert all unactivated edits (including those that have been saved to disk), or all edits made since the last save operation. This command does not release the edit session.

In the event of an error, the command returns a WLSTException.

3.7.20.2 Syntax

undo([unactivatedChanges], [defaultAnswer])

Argument	Definition
unactivatedChanges	Optional. Boolean value specifying whether to undo all unactivated changes, including edits that have been saved to disk. This argument defaults to false, indicating that all edits since the last save operation are reverted.
defaultAnswer	Optional. Default response, if you would prefer not to be prompted at the command line. Valid values are y and n. This argument defaults to null, and WLST prompts you for a response.

3.7.20.3 Example

The following example reverts all changes since the last save operation. WLST prompts for verification before reverting.

```
wls:/mydomain/edit !> undo()
Sure you would like to undo your changes? (y/n)
y
Discarded your in-memory changes successfully.
wls:/mydomain/edit>
```

The following example reverts all unactivated changes. WLST prompts for verification before reverting.

```
wls:/mydomain/edit !> undo('true')
Sure you would like to undo your changes? (y/n)
y
Discarded all your changes successfully.
wls:/mydomain/edit>
```

3.7.21 validate

Command Category: Editing Commands

Use with WLST: Online

3.7.21.1 Description

Validates the changes that have been made but have not yet been saved. This command enables you to verify that all changes are valid before saving them.

In the event of an error, the command returns a WLSTException.

3.7.21.2 Syntax

```
validate()
```

3.7.21.3 Example

The following example validates all changes that have been made but have not yet been saved.

```
wls:/mydomain/edit !> validate()
Validating changes ...
Validated the changes successfully
```

3.8 Information Commands

Use the WLST information commands, listed in Table 3–8, to interrogate domains, servers, and variables, and provide configuration bean, runtime bean, and WLST-related information.

Table 3–8   Information Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
addListener	Add a JMX listener to the specified MBean.	Online
configToScript	Convert an existing server configuration (`config` directory) to an executable WLST script	Online or Offline
dumpStack	Display stack trace from the last exception that occurred while performing a WLST action, and reset the stack trace.	Online or Offline
dumpVariables	Display all variables used by WLST, including their name and value.	Online or Offline
find	Find MBeans and attributes in the current hierarchy.	Online
getConfigManager	Return the latest `ConfigurationManagerBean` MBean which manages the change process.	Online
getMBean	Return the MBean by browsing to the specified path.	Online
getMBI	Return the `MBeanInfo` for the specified `MBeanType` or the `cmo` variable.	Online
getPath	Return the MBean path for the specified MBean instance.	Online
listChildTypes	List all the children MBeans that can be created or deleted for the cmo type.	Online
lookup	Look up the specified MBean.	Online
ls	List all child beans and/or attributes for the current configuration or runtime bean.	Online or Offline
man	Display help from `MBeanInfo` for the current MBean or its specified attribute.	Online
redirect	Redirect WLST output to the specified filename.	Online
removeListener	Remove a listener that was previously defined.	Online

Table 3–8 (Cont.) Information Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
showListeners	Show all listeners that are currently defined.	Online
startRecording	Record all user interactions with WLST; useful for capturing commands to replay.	Online or Offline
state	Returns a map of servers or clusters and their state using Node Manager.	Online
stopRecording	Stop recording WLST commands.	Online or Offline
stopRedirect	Stop redirection of WLST output to a file.	Online or Offline
storeUserConfig	Create a user configuration file and an associated key file.	Online
threadDump	Display a thread dump for the specified server.	Online or Offline
viewMBean	Display information about an MBean, such as the attribute names and values, and operations.	Online
writeIniFile	Convert WLST definitions and method declarations to a Python (.py) file.	Online or Offline

3.8.1 addListener

Command Category: Information Commands

Use with WLST: Online

3.8.1.1 Description

Adds a JMX listener to the specified MBean. Any changes made to the MBean are reported to standard out and/or are saved to the specified configuration file.

In the event of an error, the command returns a `WLSTException`.

3.8.1.2 Syntax

addListener(mbean, [attributeNames], [logFile], [listenerName])

Argument	Definition
mbean	Name of the MBean or MBean object to listen on.
attributeNames	Optional. Comma-separated list of all attribute names on which you would like to add a JMX listener. This argument defaults to null, and adds a JMX listener for all attributes.
logFile	Optional. Name and location of the log file to which you want to write listener information.This argument defaults to standard out.
listenerName	Optional. Name of the JMX listener. This argument defaults to a WLST-generated name.

3.8.1.3 Example

The following example defines a JMX listener on the `cmo` MBean for the `Notes` and `ArchiveConfigurationCount` attributes. The listener is named `domain-listener` and is stored in `./listeners/domain.log`.

```
wls:/mydomain/serverConfig> addListener(cmo, "Notes,ArchiveConfigurationCount",
"./listeners/domain.log","domain-listener")
```

3.8.2 configToScript

Command Category: Information Commands

Use with WLST: Online or Offline

Converts an existing server configuration (`config` directory) to an executable WLST script. You can use the resulting script to re-create the resources on other servers.

The configToScript command creates the following files:

- A WLST script that contains the commands needed to recreate the configuration.

- A properties file that contains domain-specific values. You can update the values in this file to create new domains that are similar to the original configuration.

- A user configuration file and an associated key file to store encrypted attributes. The user configuration file contains the encrypted information. The key file contains a secret key that is used to encrypt and decrypt the encrypted information.

When you run the generated script:

- If a server is currently running, WLST will try to connect using the values in the properties file and then run the script commands to create the server resources.

- If no server is currently running, WLST will start a server with the values in the properties file, run the script commands to create the server resources, and shutdown the server. This may cause WLST to exit from the command shell.

In the event of an error, the command returns a `WLSTException`.

3.8.2.1 Syntax

```
configToScript([configPath], [pyPath], [overwrite], [propertiesFile],
[createDeploymentScript])
```

Argument	Definition
configPath	Optional. Path to the `domain` directory that contains the configuration that you want to convert. This argument defaults to the directory from which you start WLST (`./`).
pyPath	Optional. Path and filename to which you want to write the converted WLST script. This argument defaults to `./config/config.py`.
overwrite	Optional. Boolean value specifying whether the script file should be overwritten if it already exists. This argument defaults to `true`, indicating that the script file is overwritten.
propertiesFile	Optional. Path to the directory in which you want WLST to write the properties files. This argument defaults to the pathname specified for the `scriptPath` argument.
createDeploymentScript	Optional. Boolean value specifying whether WLST creates a script that performs deployments only. This argument defaults to `false`, indicating that a deployment script is not created.

3.8.2.2 Example

The following example converts the configuration to a WLST script `config.py`. By default, the configuration file is loaded from `./config`, the script file is saved to

```
.config/config.py, and the properties files is saved to
.config/config.py.properties.
```

```
wls:/offline> configToScript()
configToScript is loading configuration from c:\Oracle\Middleware
\user_projects\domains\wls\config\config.xml ...
Completed configuration load, now converting resources to wlst script...
configToScript completed successfully
The WLST script is written to c:\Oracle\Middleware
\user_projects\domains\wls\config\config.py
and the properties file associated with this script is written to c:\Oracle\
Middleware\user_projects\domains\wls\config\config.py.properties
wls:/offline>
```

The following example converts server resources configured in the file
`c:\Oracle\Middleware\user_projects\domains\mydomain\config`
directory to a WLST script `c:\Oracle\Middleware\myscripts\config.py`.

```
wls:/offline> configToScript('c:/Oracle/Middleware/user_projects/domains
/mydomain','c:/Oracle/Middleware/myscripts')
configToScript is loading configuration from c:\Oracle\Middleware
\user_projects\domains\mydomain\config\config.xml ...
Completed configuration load, now converting resources to wlst script...
configToScript completed successfully
The WLST script is written to c:\Oracle\Middleware\myscripts\config.py
and the properties file associated with this script is written to
c:\Oracle\Middlware\mydomain\config.py.properties
wls:/offline>
```

3.8.3 dumpStack

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.3.1 Description

Displays the stack trace from the last exception that occurred while performing a WLST action, and resets the stack trace.

If successful, the dumpstack command returns the Throwable object. In the event of an error, the command returns a WLSTException.

3.8.3.2 Syntax

```
dumpStack()
```

3.8.3.3 Example

This example displays the stack trace.

```
wls:/myserver/serverConfig> dumpStack()
com.bea.plateng.domain.script.jython.WLSTException: java.lang.reflect.Invocation
TargetException
...
```

3.8.4 dumpVariables

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.4.1 Description

Displays all the variables used by WLST, including their name and value. In the event of an error, the command returns a `WLSTException`.

3.8.4.2 Syntax

```
dumpVariables()
```

3.8.4.3 Example

This example displays all the current variables and their values.

```
wls:/mydomain/serverConfig> dumpVariables()
adminHome   weblogic.rmi.internal.BasicRemoteRef - hostID:
    '-1 108080150904263937S:localhost:[7001,8001,-1,-1,-1,-1,-1]:
    mydomain:AdminServer', oid: '259', channel: 'null'
cmgr   [MBeanServerInvocationHandler]com.bea:Name=ConfigurationManager,
    Type=weblogic.management.mbeanservers.edit.ConfigurationManagerMBean
cmo   [MBeanServerInvocationHandler]com.bea:Name=mydomain,Type=Domain
connected true
domainName mydomain
...
wls:/mydomain/serverConfig>
```

3.8.5 find

Command Category: Information Commands

Use with WLST: Online

3.8.5.1 Description

Finds MBeans and attributes in the current hierarchy.

WLST returns the pathname to the MBean that stores the attribute and/or attribute type, and its value. If `searchInstancesOnly` is set to `false`, this command also searches the MBeanType paths that are not instantiated in the server, but that can be created. In the event of an error, the command returns a `WLSTException`.

3.8.5.2 Syntax

```
find([name], [type], [searchInstancesOnly])
```

Argument	Definition
name	Optional. Name of the attribute to find.
type	Optional. Type of the attribute to find.
searchInstancesOnly	Optional. Boolean value specifying whether to search registered instances only or to also search `MBeanTypes` paths that are not instantiated in the server, but that can be created. This argument defaults to `true`, indicating only the registered instances will be searched.

3.8.5.3 Example

The following example searches for an attribute named `javaCompiler` in the current configuration hierarchy.

```
wls:/mydomain/serverConfig> find(name = 'JavaCompiler')
Finding 'JavaCompiler' in all registered MBean instances ...
```

```
/Servers/AdminServer                        JavaCompilerPreClassPath    null
/Servers/AdminServer                        JavaCompiler                java
/Servers/AdminServer                        JavaCompilerPostClassPath   null
wls:/mydomain/serverConfig>
```

The following example searches for an attribute of type JMSRuntime in the current configuration hierarchy.

```
wls:/mydomain/serverRuntime> find(type='JMSRuntime')
Finding MBean of type 'JMSRuntime' in all the instances ...
/JMSRuntime/AdminServer.jms
wls:/mydomain/serverRuntime>
```

The following example searches for an attribute named execute in the current configuration hierarchy. The searchInstancesOnly argument is set to false, indicating to also search MBeanTypes that are not instantiated in the server.

```
wls:/mydomain/serverConfig> find(name='execute', searchInstancesOnly='false')
Finding 'execute' in all registered MBean instances ...
/Servers/AdminServer     ExecuteQueues [Ljavax.management.ObjectName;@1aa7dbc
/Servers/AdminSever      Use81StyleExecuteQueues                         false
Now finding 'execute' in all MBean Types that can be instantiated ...
/Servers                                    ExecuteQueues
/Servers                                    Use81StyleExecuteQueues
wls:/mydomain/serverConfig>
```

3.8.6 getConfigManager

Command Category: Information Commands

Use with WLST: Online

3.8.6.1 Description

Returns the latest ConfigurationManager MBean which manages the change process. You can then invoke methods to manage configuration changes across a WebLogic domain. In the event of an error, the command returns a WLSTException.

3.8.6.2 Syntax

```
getConfigManager()
```

3.8.6.3 Example

The following example returns the latest ConfigurationManagerBean MBean and stores it in a cm variable.

```
wls:/mydomain/serverConfig> cm=getConfigManager()
wls:/mydomain/serverConfig> cm.getType()
'weblogic.management.mbeanservers.edit.ConfigurationManagerMBean'
```

3.8.7 getMBean

Command Category: Information Commands

Use with WLST: Online

3.8.7.1 Description

Returns the MBean by browsing to the specified path. In the event of an error, the command returns a WLSTException.

> **Note:** No exception is thrown if the MBean is not found.

3.8.7.2 Syntax

```
getMBean(mbeanPath)
```

Argument	Definition
mbeanPath	Path name to the MBean in the current hierarchy.

3.8.7.3 Example

The following example returns the MBean specified by the path.

```
wls:/mydomain/edit !> com=getMBean('Servers/myserver/COM/myserver')
wls:/mydomain/edit !> com.getType()
'Server'
```

3.8.8 getMBI

Command Category: Information Commands

Use with WLST: Online

3.8.8.1 Description

Returns the MBeanInfo for the specified MBeanType or the cmo variable. In the event of an error, the command returns a WLSTException.

3.8.8.2 Syntax

```
getMBI([mbeanType])
```

Argument	Definition
mbeanType	Optional. MBeanType for which the MBeanInfo is displayed.

3.8.8.3 Example

The following example gets the MBeanInfo for the specified MBeanType and stores it in the variable svrMbi.

```
wls:/mydomain/serverConfig>
svrMbi=getMBI('weblogic.management.configuration.ServerMBean')
```

3.8.9 getPath

Command Category: Information Commands

Use with WLST: Online

3.8.9.1 Description

Returns the MBean path for the specified MBean instance or ObjectName for the MBean in the current tree. In the event of an error, the command returns a WLSTException.

3.8.9.2 Syntax

```
getPath(mbean)
```

Argument	Definition
mbean	MBean instance or ObjectName for the MBean in the current tree for which you want to return the MBean path.

3.8.9.3 Example

The following example returns the MBean specified by the path.

```
wls:/mydomain/edit !> path=getPath('com.bea:Name=myserver,Type=Server')
wls:/mydomain/edit !> print path
'Servers/myserver'
```

3.8.10 listChildTypes

Command Category: Information Commands

Use with WLST: Online

3.8.10.1 Description

Lists all the child MBeans that can be created or deleted for the cmo. The cmo variable specifies the configuration bean instance to which you last navigated using WLST. For more information about the cmo variable, see "Changing the Current Management Object" in *Oracle WebLogic Scripting Tool*.

In the event of an error, the command returns a WLSTException.

3.8.10.2 Syntax

```
listChildTypes([parent])
```

Argument	Definition
parent	Optional. Parent type for which you want the children types listed.

3.8.10.3 Example

The following example lists the children MBeans that can be created or deleted for the cmo type.

```
wls:/mydomain/serverConfig> listChildTypes()
AppDeployments
BridgeDestinations
CachingRealms
Clusters
...
wls:/mydomain/serverConfig>
```

3.8.11 lookup

Command Category: Information Commands

Use with WLST: Online

3.8.11.1 Description

Looks up the specified MBean. The MBean must be a child of the current MBean. In the event of an error, the command returns a WLSTException.

3.8.11.2 Syntax

```
lookup(name, [childMBeanType])
```

Argument	Definition
name	Name of the MBean that you want to lookup.
childMBeanType	Optional. The type of the MBean that you want to lookup.

3.8.11.3 Example

The following example looks up the specified server, myserver, and stores the returned stub in the sbean variable.

```
wls:/mydomain/serverConfig> sbean=lookup('myserver','Server')
wls:/mydomain/serverConfig> sbean.getType()
'Server'
wls:/mydomain/serverConfig>
```

3.8.12 ls

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.12.1 Description

Lists the attributes, operations, and child management objects of the specified management object.

In the event of an error, the command returns a WLSTException.

By default, the output is returned as a string and is arranged in three columns:

- The first column displays a set of codes that describe the listed item. See Table 3–9.

- The second column displays the item name.

- When the item is an attribute, the third column displays the attribute value. If an attribute is encrypted, the third column displays asterisks instead of the value. (See "Writing and Reading Encrypted Configuration Values" in *Oracle WebLogic Scripting Tool*.)

- When the item is an operation, the third column uses the following pattern to display the operation's return type and input parameters: *returnType*: *parameterType*(*parameterName*)

Table 3–9   ls Command Output Information

Code	Description
d	Indicates that the item is a child management object. Like a directory in a UNIX or Windows file system, you can use the cd command to make the child object the current management object.
r	Indicates that the item is a child management object or an attribute that is readable, assuming that current user has been given read permission by the security realm's policies. (See "Default Security Policies for MBeans" in the *Oracle WebLogic Server MBean Reference*.)
w	Indicates that the item is an attribute that is writable, assuming that current user has been given write permission by the security realm's policies. (See "Default Security Policies for MBeans" in the *Oracle WebLogic Server MBean Reference*.)

Table 3–9   (Cont.) ls Command Output Information

Code	Description
x	Indicates that the item is an operation that can be executed, assuming that current user has been given execute permission by the security realm's policies. (See "Default Security Policies for MBeans" in the *Oracle WebLogic Server MBean Reference*.)

By default, the output lists all attributes, operations, and child management objects of the current management object. To filter the output or to see a list for a different management object, you can specify a command argument.

> **Note:**   As a performance optimization, when using WLST offline, WebLogic Server does not store most of its default values in the configuration files for the WebLogic domain. In some cases, this optimization prevents entire management objects from being displayed by WLST offline (because WebLogic Server has never written the corresponding XML elements to the domain configuration files). For example, if you never modify the default logging severity level for a WebLogic domain while the domain is active, WLST offline will not display the Log management object for the domain.
>
> If you want to change the default value of attributes whose management object is not displayed by WLST offline, you must first use the create command to create the management object. Then you can cd to the management object and change the attribute value. See Section 3.7.4, "create".

3.8.12.2 Syntax

```
ls( [ a | c | o ] [ moPath ])

ls( [ moPath ] returnMap [ returnType ] )
```

Argument	Definition
a	Optional. Displays only the attributes of the specified management object (suppresses the display of other items).
c	Optional. Displays only the child management objects of the specified management object (suppresses the display of other items).
o	Optional. Displays only the operations that can be invoked on the specified management object (suppresses the display of other items).
	This argument is only applicable for WLST online.

Argument	Definition
moPath	Optional. Path name to the management object for which you want to list attributes, operations, and child management objects.
	You can specify a pathname that is relative to your current location in the hierarchy or an absolute pathname.
	With WLST offline, use the forward-slash character (/) to specify the root of the configuration document.
	With WLST online, you can list the contents of MBeans in any management hierarchy (see Section 3.11, "Tree Commands"). Use the following syntax to specify the root of a hierarchy:
	root-name:/
	For example, to list the root of the server runtime hierarchy:
	ls('serverRuntime:/')
	If you do not specify this argument, the command lists items for the current management object.
returnMap	Optional. Boolean value that determines whether the command returns values as a map. This argument defaults to false, which causes this command to return a String.
returnType	Optional. Controls the output returned in the map. Specify a, c, or o, which filter the output as described at the top of this table.
	This argument is valid only if returnMap is set to true. This argument defaults to c.

3.8.12.3 Example

The following example displays all the child configuration beans, and attribute names and values for the examples domain, which has been loaded into memory, in WLST offline mode:

```
wls:/offline/mydomain > ls()
dr--    AppDeployments
dr--    BridgeDestinations
dr--    Clusters
dr--    CustomResources
dr--    DeploymentConfiguration
dr--    Deployments
dr--    EmbeddedLDAP
dr--    ErrorHandlings
dr--    FileStores
dr--    InternalAppDeployments
dr--    InternalLibraries
dr--    JDBCDataSourceFactories
dr--    JDBCStores
dr--    JDBCSystemResources
dr--    JMSBridgeDestinations
dr--    JMSInteropModules
dr--    JMSServers
dr--    JMSSystemResources
dr--    JMX
...
wls:/offline/examples>
```

The following example displays all the attribute names and values in DomainMBean:

```
wls:/mydomain/serverConfig> ls('a')
-r--    AdminServerName                       AdminServer
-r--    AdministrationMBeanAuditingEnabled    false
```

```
-r--    AdministrationPort                        9002
-r--    AdministrationPortEnabled                 false
-r--    AdministrationProtocol                    t3s
-r--    ArchiveConfigurationCount                 0
-r--    ClusterConstraintsEnabled                 false
-r--    ConfigBackupEnabled                       false
-r--    ConfigurationAuditType                    none
-r--    ConfigurationVersion                      9.0.0.0
-r--    ConsoleContextPath                        console
-r--    ConsoleEnabled                            true
-r--    ConsoleExtensionDirectory                 console-ext
-r--    DomainVersion                             9.0.0.0
-r--    LastModificationTime                      0
-r--    Name                                      basicWLSDomain
-r--    Notes                                     null
-r--    Parent                                    null
-r--    ProductionModeEnabled                     false
-r--    RootDirectory                             .
-r--    Type                                      Domain
wls:/mydomain/serverConfig>
```

The following example displays all the child beans and attribute names and values in `Servers` MBean:

```
wls:/mydomain/serverConfig> ls('Servers')
dr--    AdminServer
```

The following example displays the attribute names and values for the specified MBean path and returns the information in a map:

```
wls:/mydomain/serverConfig> svrAttrList = ls('edit:/Servers/myserver', 'true',
'a')
-rw-    AcceptBacklog                             50
-rw-    AdminReconnectIntervalSeconds             10
-rw-    AdministrationPort                        9002
-rw-    AdministrationProtocol                    t3s
-rw-    AutoKillIfFailed                          false
-rw-    AutoMigrationEnabled                      false
-rw-    AutoRestart                               true
-rw-    COMEnabled                                false
-rw-    ClasspathServletDisabled                  false
-rw-    ClientCertProxyEnabled                    false
-rw-    Cluster                                   null
-rw-    ClusterRuntime                            null
-rw-    ClusterWeight                             100
wls:/mydomain/serverConfig>
```

3.8.13 man

Command Category: Information Commands

Use with WLST: Online

3.8.13.1 Description

Displays help from `MBeanInfo` for the current MBean or its specified attribute. In the event of an error, the command returns a `WLSTException`.

3.8.13.2 Syntax

```
man([attrName])
```

Argument	Definition
attrName	Optional. MBean attribute name for which you would like to display help. If not specified, WLST displays helps for the current MBean.

3.8.13.3 Example

The following example displays help from `MBeanInfo` for the `ServerMBean` bean.

```
wls:/mydomain/serverConfig> man('Servers')
dynamic : true
creator : createServer
destroyer : destroyServer
description : <p>Returns the ServerMBeans representing the servers that have been
configured to be part of this domain.</p>
descriptorType : Attribute
Name : Servers
interfaceClassName : [Lweblogic.management.configuration.ServerMBean;
displayName : Servers
relationship : containment
```

3.8.14 redirect

Command Category: Information Commands

Use with WLST: Online

3.8.14.1 Description

Redirects WLST information, error, and debug messages to the specified filename. Also redirects the output of the `dumpStack()` and `dumpVariables()` commands to the specified filename.

In the event of an error, the command returns a `WLSTException`.

3.8.14.2 Syntax

```
redirect(outputFile, [toStdOut])
```

Argument	Definition
outputFile	Name of the file to which you want to record the WLST commands. The filename can be absolute or relative to the directory from which you started WLST.
toStdOut	Optional. Boolean value specifying whether the output should be sent to `stdout`. This argument defaults to `true`, indicating that the output will be sent to `stdout`.

3.8.14.3 Example

The following example begins redirecting WLST output to the `logs/wlst.log` file:

```
wls:/mydomain/serverConfig> redirect('./logs/wlst.log')
```

3.8.15 removeListener

Command Category: Information Commands

Use with WLST: Online

3.8.15.1 Description

Removes a listener that was previously defined. If you do not specify an argument, WLST removes all listeners defined for all MBeans. For information about setting a listener, see Section 3.8.1, "addListener".

In the event of an error, the command returns a `WLSTException`.

3.8.15.2 Syntax

```
removeListener([mbean], [listenerName])
```

Argument	Definition
mbean	Optional. Name of the MBean or MBean object for which you want to remove the previously defined listeners.
listenerName	Optional. Name of the listener to be removed.

3.8.15.3 Example

The following example removes the listener named `mylistener`.

```
wls:/mydomain/serverConfig> removeListener(listenerName="mylistener")
```

3.8.16 showListeners

Command Category: Information Commands

Use with WLST: Online

3.8.16.1 Description

Shows all listeners that are currently defined. For information about setting a listener, see Section 3.8.1, "addListener".

In the event of an error, the command returns a `WLSTException`.

3.8.16.2 Syntax

```
showListeners()
```

3.8.16.3 Example

The following example shows all listeners that are currently defined.

```
wls:/mydomain/serverConfig> showListeners()
```

3.8.17 startRecording

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.17.1 Description

Records all user interactions with WLST. This command is useful for capturing commands for replay.

In the event of an error, the command returns a `WLSTException`.

This command cannot be used when you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*.

3.8.17.2 Syntax

```
startRecording(recordFile, [recordAll])
```

Argument	Definition
recordFile	Name of the file to which you want to record the WLST commands. The filename can be absolute or relative to the directory from which you invoked WLST.
recordAll	Optional. Boolean value specifying whether to capture all user interactions in the file. This argument defaults to `false`, indicating that only WLST commands are captured, and not WLST command output.

3.8.17.3 Example

The following example begins recording WLST commands in the `record.py` file:

```
wls:/mydomain/serverConfig> startRecording('c:/myScripts/record.py')
Starting recording to c:/myScripts/record.py
wls:/mydomain/serverConfig>
```

3.8.18 state

Command Category: Information Commands

Use with WLST: Online

3.8.18.1 Description

Using Node Manager, returns a map of servers or clusters and their state. Node Manager must be running.

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

In the event of an error, the command returns a `WLSTException`.

3.8.18.2 Syntax

```
state(name, [type])
```

Argument	Definition
name	Name of the server or cluster for which you want to retrieve the current state.
type	Optional. Type, `Server` or `Cluster`. This argument defaults to `Server`. When returning the state of a cluster, you must set this argument explicitly to `Cluster`, or the command will fail.

3.8.18.3 Example

The following example returns the state of the Managed Server, `managed1`.

```
wls:/mydomain/serverConfig> state('managed1','Server')
Current state of "managed1": SUSPENDED
wls:/mydomain/serverConfig>
```

The following example returns the state of the cluster, `mycluster`.

```
wls:/mydomain/serverConfig> state('mycluster','Cluster')
There are 3 server(s) in cluster: mycluster

States of the servers are
```

```
                    MServer1---SHUTDOWN
                    MServer2---SHUTDOWN
                    MServer3---SHUTDOWN
                    wls:/mydomain/serverConfig>
```

3.8.19 stopRecording

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.19.1 Description

Stops recording WLST commands. For information about starting a recording, see
Section 3.8.17, "startRecording".

In the event of an error, the command returns a `WLSTException`.

3.8.19.2 Syntax

```
stopRecording()
```

3.8.19.3 Example

The following example stops recording WLST commands.

```
wls:/mydomain/serverConfig> stopRecording()
Stopping recording to c:\myScripts\record.py
wls:/mydomain/serverConfig>
```

3.8.20 stopRedirect

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.20.1 Description

Stops the redirection of WLST output to a file, if redirection is in progress.

In the event of an error, the command returns a `WLSTException`.

3.8.20.2 Syntax

```
stopRedirect()
```

3.8.20.3 Example

The following example stops the redirection of WLST output to a file:

```
wls:/mydomain/serverConfig> stopRedirect()
WLST output will not be redirected to myfile.txt any more
```

3.8.21 storeUserConfig

Command Category: Information Commands

Use with WLST: Online

3.8.21.1 Description

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

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

In the event of an error, the command returns a `WLSTException`.

3.8.21.2 Syntax

```
storeUserConfig([userConfigFile], [userKeyFile], [nm])
```

Argument	Definition
userConfigFile	Optional. Name of the file to store the user configuration. The pathname can be absolute or relative to the file-system directory from which you started WLST.
	If you do not specify this argument, the command stores the file in your home directory as determined by your JVM. The location of the home directory depends on the SDK and type of operating system on which WLST is running.The default filename is based on the following pattern:
	username-WebLogicConfig.properties
	where *username* is the user name that you used to log in to the operating system.
	The command also prints to standard out the location in which it created the file.
userKeyFile	Optional. Name of the file to store the key information that is associated with the user configuration file that you specify. The pathname can be absolute or relative to the file-system directory from which you started WLST.
	If you do not specify this argument, the command stores the file in your home directory as determined by your JVM. The location of the home directory depends on the SDK and type of operating system on which WLST is running. The default filename is based on the following pattern:
	username-WebLogicKey.properties
	where *username* is the user name that you used to log in to the operating system.
	The command also prints to standard out the location in which it created the file.
nm	Optional. Boolean value specifying whether to store the username and password for Node Manager or WebLogic Server. If set to true, the Node Manager username and password is stored. This argument default to `false`.

3.8.21.3 Example

The following example creates and stores a user configuration file and key file in the default location.

```
wls:/mydomain/serverConfig> storeUserConfig()
Creating the key file can reduce the security of your system if it is not kept in
a secured location after it is created. Do you want to create the key file? y or n
y
The username and password that were used for this current WLS connection are
stored in C:\Documents and Settings\pat\pat-WebLogicConfig.properties
```

and C:\Documents and Settings\pat\pat-WebLogicKey.properties.

The following example creates and stores a user configuration file and key file in the specified locations.

```
wls:/mydomain/serverConfig> storeUserConfig('c:/myFiles/myuserconfigfile.secure',
'c:/myFiles/myuserkeyfile.secure')
Creating the key file can reduce the security of your system if it is not kept in
a secured location after it is created. Do you want to create the key file? y or n
y
The username and password that were used for this current WLS connection are
stored in c:/myFiles/mysuserconfigfile.secure and c:/myFiles/myuserkeyfile.secure
wls:/mydomain/serverConfig>
```

3.8.22 threadDump

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.22.1 Description

Displays a thread dump for the specified server. In the event of an error, the command returns a WLSTException.

3.8.22.2 Syntax

threadDump([writeToFile], [fileName], [serverName])

Argument	Definition
writeToFile	Optional. Boolean value specifying whether to save the output to a file. This argument defaults to true, indicating that output is saved to a file.
fileName	Optional. Name of the file to which the output is written. The filename can be absolute or relative to the directory where WLST is running. This argument defaults to Thread_Dump_*serverName* file, where *serverName* indicates the name of the server. This argument is valid only if *writeToFile* is set to true.
serverName	Optional. Server name for which the thread dump is requested. This argument defaults to the server to which WLST is connected.
	If you are connected to an Administration Server, you can display a thread dump for the Administration Server and any Managed Server that is running in the WebLogic domain. If you are connected to a Managed Server, you can only display a thread dump for that Managed Server.

3.8.22.3 Example

The following example displays the thread dump for the current server and saves the output to the Thread_Dump_*serverName* file.

```
wls:/mydomain/serverConfig> threadDump()
```

The following example displays the thread dump for the server managedServer. The information is not saved to a file.

```
wls:/mydomain/serverConfig> threadDump(writeToFile='false',
serverName='managedServer')
```

3.8.23 viewMBean

Command Category: Information Commands

Use with WLST: Online

3.8.23.1 Description

Displays information about an MBean, such as the attribute names and values, and operations. In the event of an error, the command returns a `WLSTException`.

3.8.23.2 Syntax

```
viewMBean(mbean)
```

Argument	Definition
mbean	MBean for which you want to display information.

3.8.23.3 Example

The following example displays information about the current MBean, `cmo`.

```
wls:/mydomain/serverConfig> cmo.getType()
'Domain'
wls:/mydomain/serverConfig> viewMBean(cmo)
Attribute Names and Values
--------------------------
XMLEntityCaches    null
Targets    javax.management.ObjectName[com.bea
:Name=MedRecJMSServer,Type=JMSServer,
    com.bea:Name=WSStoreForwardInternalJMSServerMedRecServer,Type=JMSServer,
    com.bea:Name=MedRecWseeJMSServer,Type=JMSServer,
    com.bea:Name=PhysWSEEJMSServer,Type=JMSServer,
    com.bea:Name=MedRecSAFAgent,Type=SAFAgent,
    com.bea:Name=AdminServer,Type=Server]
RootDirectory                               .
EmbeddedLDAP                                com.bea:Name=OOTB_medrec,Type=EmbeddedLDAP
RemoteSAFContexts  null
Libraries    javax.management.ObjectName[com.bea
...
wls:/mydomain/serverConfig>
```

3.8.24 writeIniFile

Command Category: Information Commands

Use with WLST: Online

3.8.24.1 Description

Converts WLST definitions and method declarations to a Python (`.py`) file to enable advanced users to import them as a Jython module. After importing, the definitions and method declarations are available to other Jython modules and can be accessed directly using Jython syntax. For more information, see "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*.

In the event of an error, the command returns a `WLSTException`.

3.8.24.2 Syntax

```
writeIniFile(filePath)
```

Argument	Definition
filePath	Full pathname to the file that you want to save the converted information.

3.8.24.3 Example

The following example converts WLST to a Python file named `wl.py`.

```
wls:/offline> writeIniFile("wl.py")
The Ini file is successfully written to wl.py
wls:/offline>
```

3.9 Life Cycle Commands

Use the WLST life cycle commands, listed in Table 3–10, to manage the life cycle of a server instance.

For more information about the life cycle of a server instance, see "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown for Oracle WebLogic Server*.

Table 3–10   Life Cycle Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
migrate	Migrate services to a target server within a cluster.	Online
resume	Resume a server instance that is suspended or in ADMIN state.	Online
shutdown	Gracefully shut down a running server instance or cluster.	Online
start	Start a Managed Server instance or a cluster using Node Manager.	Online
startServer	Start the Administration Server.	Online or Offline
suspend	Suspend a running server.	Online

3.9.1 migrate

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.1.1 Description

Migrates the specified services (JTA, JMS, or Server) to a targeted server within a cluster. In the event of an error, the command returns a `WLSTException`.

For information about migrating services, see "Service Migration" in *Using Clusters for Oracle WebLogic Server*.

3.9.1.2 Syntax

```
migrate(sname, destinationName, [sourceDown], [destinationDown], [migrationType])
```

Argument	Definition
sname	Name of the server from which the services should be migrated.

Argument	Definition
destinationName	Name of the machine or server to which you want to migrate the services.
sourceDown	Optional. Boolean value specifying whether the source server is down. This argument defaults to `true`, indicating that the source server is not running.
	When migrating JTA services, the *sourceDown* argument is ignored, if specified, and defaults to `true`. The source server **must** be down in order for the migration of JTA services to succeed.
destinationDown	Optional. Boolean value specifying whether the destination server is down. This argument defaults to `false`, indicating that the destination server is running.
	If the destination is not running, and you do not set this argument to `true`, WLST returns a `MigrationException`.
	When migrating JMS-related services to a non-running server instance, the server instance will activate the JMS services upon the next startup. When migrating the JTA Transaction Recovery Service to a non-running server instance, the target server instance will assume recovery services when it is started.
migrationType	Optional. Type of service(s) that you want to migrate. Valid values include:
	■ `jms`—Migrate JMS-related services (JMS server, SAF agent, path service, and the WebLogic persistent store) only.
	■ `jta`—Migrate JTA services only.
	■ `server`—Migrate Server services only.
	■ `all`—Migrate all JTA and JMS services.
	This argument defaults to `all`.

3.9.1.3 Example

The following example migrates all JMS and JTA services on `server1` to the server `server2`. The boolean arguments specify that the source server is down and the destination server is running.

```
wls:/mydomain/edit !> migrate('server1','server2', 'true', 'false', 'all')
Migrating all JMS and JTA services from 'server1' to destination 'server2' ...
wls:/mydomain/edit !>
```

The following example migrates all Server services on `server1` to the server `server2`. The boolean arguments specify that the source server is down and the destination server is running.

```
wls:/mydomain/edit !> migrate('server1','server2', 'true', 'false', 'Server')
Migrating singleton server services from 'server1' to machine 'server2'...
wls:/mydomain/edit !>
```

3.9.2 resume

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.2.1 Description

Resumes a server instance that is suspended or in `ADMIN` state. This command moves a server to the `RUNNING` state. For more information about server states, see

"Understanding Server Life Cycle" in *Managing Server Startup and Shutdown for Oracle WebLogic Server*.

In the event of an error, the command returns a `WLSTException`.

3.9.2.2 Syntax

```
resume([sname], [block])
```

Argument	Definition
sname	Name of the server to resume. This argument defaults to the server to which WLST is currently connected.
block	Optional. Boolean value specifying whether WLST should block user interaction until the server is resumed. This argument defaults to `false`, indicating that user interaction is not blocked. In this case, WLST returns control to the user after issuing the command and assigns the task MBean associated with the current task to a variable that you can use to check its status. If you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*, *block* is always set to `true`.

3.9.2.3 Example

The following example resumes a Managed Server instance.

```
wls:/mydomain/serverConfig> resume('managed1', block='true')
Server 'managed1' resumed successfully.
wls:/mydomain/serverConfig>
```

3.9.3 shutdown

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.3.1 Description

Gracefully shuts down a running server instance or a cluster. The `shutdown` command waits for all the in-process work to be completed before shutting down the server or cluster.

You shut down a server to which WLST is connected by entering the `shutdown` command without any arguments.

When connected to a Managed Server instance, you only use the `shutdown` command to shut down the Managed Server instance to which WLST is connected; you cannot shut down another server while connected to a Managed Server instance.

WLST uses Node Manager to shut down a Managed Server. When shutting down a Managed Server, Node Manager must be running.

In the event of an error, the command returns a `WLSTException`.

3.9.3.2 Syntax

```
shutdown([name], [entityType], [ignoreSessions], [timeOut], [force], [block])
```

Argument	Definition
name	Optional. Name of the server or cluster to shutdown. This argument defaults to the server to which WLST is currently connected.

Argument	Definition
entityType	Optional. Type, `Server` or `Cluster`. This argument defaults to `Server`. When shutting down a cluster, you must set this argument explicitly to `Cluster`, or the command will fail.
ignoreSessions	Optional. Boolean value specifying whether WLST should drop all HTTP sessions immediately or wait for HTTP sessions to complete or timeout while shutting down. This argument defaults to `false`, indicating that all HTTP sessions must complete or timeout.
timeOut	Optional. Time (in seconds) that WLST waits for subsystems to complete in-process work and suspend themselves before shutting down the server. This argument defaults to 0 seconds, indicating that there is no timeout.
force	Optional. Boolean value specifying whether WLST should terminate a server instance or a cluster without waiting for the active sessions to complete. This argument defaults to `false`, indicating that all active sessions must complete before shutdown.
block	Optional. Boolean value specifying whether WLST should block user interaction until the server is shutdown. This argument defaults to `false`, indicating that user interaction is not blocked. In this case, WLST returns control to the user after issuing the command and assigns the task MBean associated with the current task to a variable that you can use to check its status. If you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*, *block* is always set to `true`.

3.9.3.3 Example

The following example instructs WLST to shutdown the server to which you are connected:

```
wls:/mydomain/serverConfig> shutdown()
Shutting down the admin server that you are currently connected to .......
Disconnected from weblogic server: AdminServer
```

The following example instructs WLST to wait 1000 seconds for HTTP sessions to complete or timeout (at 1000 seconds) before shutting down `myserver`:

```
wls:/mydomain/serverConfig> shutdown('myserver','Server','false',1000,
block='false')
```

The following example instructs WLST to drop all HTTP sessions immediately while connected to a Managed Server instance:

```
wls:/mydomain/serverConfig> shutdown('MServer1','Server','true',1200)
Shutting down a managed server that you are connected to ...
Disconnected from weblogic server: MServer1
```

The following example instructs WLST to shutdown the cluster `mycluster`:

```
wls:/mydomain/serverConfig> shutdown('mycluster','Cluster')
Shutting down the cluster with name mycluster
Shutdown of cluster mycluster has been issued, please
refer to the logs to check if the cluster shutdown is successful.
Use the state(<server-name>) or state(<cluster-name>,"Cluster")
to check the status of the server or cluster
wls:/mydomain/serverConfig> state('mycluster','Cluster')
There are 3 server(s) in cluster: mycluster

States of the servers are
```

```
MServer1---SHUTDOWN
MServer2---SHUTDOWN
MServer3---SHUTDOWN
wls:/mydomain/serverConfig>
```

3.9.4 start

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.4.1 Description

Starts a Managed Server instance or a cluster using Node Manager. WLST must be connected to the Administration Server and Node Manager must be running.

For more information about WLST commands used to connect to and use Node Manager, see Section 3.10, "Node Manager Commands".

In the event of an error, the command returns a `WLSTException`.

3.9.4.2 Syntax

start(name, [type], [url], [block])

Argument	Definition
name	Name of the Managed Server or cluster to start.
type	Optional. Type, `Server` or `Cluster`. This argument defaults to `Server`. When starting a cluster, you must set this argument explicitly to `Cluster`, or the command will fail.
url	Optional. Listen address and listen port of the server instance, specified using the following format: [*protocol*://]*listen-address:listen-port*. If not specified, this argument defaults to t3://localhost:7001.
block	Optional. Boolean value specifying whether WLST should block user interaction until the server or cluster is started. This argument defaults to `false`, indicating that user interaction is not blocked. In this case, WLST returns control to the user after issuing the command and assigns the task MBean associated with the current task to a variable that you can use to check its status. If you are importing WLST as a Jython module, as described "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*, *block* is always set to `true`.

3.9.4.3 Example

The following example instructs Node Manager to start a Managed Server instance; the listen address is localhost and listen port is 8801. WLST returns control to the user after issuing this command, as block is set to false.

```
wls:/mydomain/serverConfig> start('myserver', 'Server', block='false')
Starting server myserver ...
Server with name myserver started successfully.
wls:/mydomain/serverConfig>
```

The following example instructs Node Manager to start a cluster. WLST block user interaction until the cluster is started, as block defaults to true.

```
wls:/mydomain/serverConfig> start('mycluster', 'Cluster')
Starting the following servers in Cluster, mycluster: MS1, MS2, MS3...
.............................................................
```

```
            All servers in the cluster mycluster are started successfully.
            wls:/mydomain/serverConfig>
```

3.9.5 startServer

Command Category: Life Cycle Commands

Use with WLST: Online or Offline

3.9.5.1 Description

Starts the Administration Server. In the event of an error, the command returns a
WLSTException.

Note: You can use startServer only to start a WebLogic
Administration Server, by running WLST from the WL_
HOME/common/bin directory. You cannot use startServer to
start an integrated WebLogic Administration Server (that is, an
Administration Server for a Fusion Middleware Suite product
installed in an ORACLE_HOME directory).

To start the Administration server for a Fusion Middleware Suite
product other than WebLogic Server, use either of the following
methods:

- Execute the server startup script for the associated WebLogic
  domain.

- Start the server using Node Manager. If you use this method,
  make sure that the startScriptEnabled property is set to
  true in Node Manager.

3.9.5.2 Syntax

```
startServer([adminServerName], [domainName], [url], [username], [password],
[domainDir], [block], [timeout], [serverLog], [systemProperties], [jvmArgs]
[spaceAsJvmArgsDelimiter])
```

Argument	Definition
adminServerName	Optional. Name of the Administration Server to start. This argument defaults to myserver.
domainName	Optional. Name of the WebLogic domain to which the Administration Server belongs. This argument defaults to mydomain.
url	Optional. URL of the Administration Server. The URL supplied with the startServer command will override the listen address and port specified in the config.xml file. If not specified on the command line or in the config.xml file, this argument defaults to t3://localhost:7001.
username	Optional. Username use to connect WLST to the server. This argument defaults to weblogic.
password	Optional. Password used to connect WLST to the server. This argument defaults to welcome1.
domainDir	Optional. Domain directory in which the Administration Server is being started. This argument defaults to the directory from which you started WLST.

Argument	Definition
block	Optional. Boolean value specifying whether WLST blocks user interaction until the server is started. When *block* is set to `false`, WLST returns control to the user after issuing the command. This argument defaults to `true`, indicating that user interaction is blocked. If you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*, *block* is always set to `true`.
timeout	Optional. Time (in milliseconds) that WLST waits for the server to start before canceling the operation. The default value is 60000 milliseconds. This argument is only applicable when *block* is set to `true`.
serverLog	Optional. Location of the server log file. This argument defaults to `stdout`.
systemProperties	Optional. System properties to pass to the server process. System properties should be specified as comma-separated name-value pairs, and the name-value pairs should be separated by equals sign (=).
jvmArgs	Optional. JVM arguments to pass to the server process. Multiple arguments can be specified, separated by commas.
spaceAsJvmArgsDel imiter	Optional. Boolean value specifying whether JVM arguments are space delimited. The default value is false.

3.9.5.3 Example

The following example starts the Administration Server named `demoServer` in the `demoDomain`.

```
wls:/offline> startServer('demoServer','demoDomain','t3://localhost:8001',
'myweblogic','wlstdomain','c:/mydomains/wlst','false', 60000,
jvmArgs='-XX:MaxPermSize=75m, -Xmx512m, -XX:+UseParallelGC')
wls:/offline>
```

3.9.6 suspend

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.6.1 Description

Suspends a running server. This command moves a server from the `RUNNING` state to the `ADMIN` state. For more information about server states, see "Understanding Server Life Cycle" in *Managing Server Startup and Shutdown for Oracle WebLogic Server*.

In the event of an error, the command returns a `WLSTException`.

3.9.6.2 Syntax

suspend([sname], [ignoreSessions], [timeOut], [force], [block])

Argument	Definition
sname	Optional. Name of the server to suspend. The argument defaults to the server to which WLST is currently connected.
ignoreSessions	Optional. Boolean value specifying whether WLST should drop all HTTP sessions immediately or wait for HTTP sessions to complete or time out while suspending. This argument defaults to `false`, indicating that HTTP sessions must complete or time out.

Argument	Definition
timeOut	Optional. Time (in seconds) the WLST waits for the server to complete in-process work before suspending the server. This argument defaults to 0 seconds, indicating that there is no timeout.
force	Optional. Boolean value specifying whether WLST should suspend the server without waiting for active sessions to complete. This argument defaults to false, indicating that all active sessions must complete before suspending the server.
block	Optional. Boolean value specifying whether WLST blocks user interaction until the server is started. This argument defaults to false, indicating that user interaction is not blocked. In this case, WLST returns control to the user after issuing the command and assigns the task MBean associated with the current task to a variable that you can use to check its status. If you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*, *block* is always set to true.

3.9.6.3 Example

The following example suspends a Managed Server instance:

```
wls:/mydomain/serverConfig> suspend('managed1')
Server 'managed1' suspended successfully.
wls:/mydomain/serverConfig>
```

3.10 Node Manager Commands

Use the WLST Node Managers commands, listed in Table 3–11, to start, shut down, restart, and monitor WebLogic Server instances.

> **Note:** Node Manager must be running before you can execute the commands within this category.

For more information about Node Manager, see "Using Node Manager" in the *Node Manager Administrator's Guide for Oracle WebLogic Server*.

Table 3–11   Node Manager Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
nm	Determine whether WLST is connected to Node Manager.	Online
nmConnect	Connect WLST to Node Manager to establish a session.	Online or Offline
nmDisconnect	Disconnect WLST from a Node Manager session.	Online or Offline
nmEnroll	Enables the Node Manager on the current computer to manage servers in a specified WebLogic domain.	Online
nmGenBootStartupProps	Generates the Node Manager property files, boot.properties and startup.properties, for the specified server.	Online
nmKill	Kill the specified server instance that was started with Node Manager.	Online or Offline

Table 3–11   (Cont.) Node Manager Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
nmLog	Return the Node Manager log.	Online or Offline
nmServerLog	Return the server output log of the server that was started with Node Manager.	Online or Offline
nmServerStatus	Return the status of the server that was started with Node Manager.	Online or Offline
nmStart	Start a server in the current WebLogic domain using Node Manager.	Online or Offline
nmVersion	Return the Node Manager version.	Online or Offline
startNodeManager	Starts Node Manager on the same computer that is running WLST.	Online or Offline
stopNodeManager	Stops Node Manager.	Online or Offline

3.10.1 nm

Command Category: Node Manager Commands

Use with WLST: Online or Offline

3.10.1.1 Description

Determines whether WLST is connected to Node Manager. Returns `true` or `false` and prints a descriptive message. Node Manager must be running before you can execute this command.

In the event of an error, the command returns a `WLSTException`.

3.10.1.2 Syntax

```
nm()
```

3.10.1.3 Example

The following example indicates that WLST is currently connected to Node Manager that is monitoring `mydomain`.

```
wls:/mydomain/serverConfig> nm()
Currently connected to Node Manager that is monitoring the domain "mydomain"
wls:/mydomain/serverConfig>
```

The following example indicates that WLST is not currently connected to Node Manager.

```
wls:/mydomain/serverConfig> nm()
Not connected to any Node Manager
wls:/mydomain/serverConfig>
```

3.10.2 nmConnect

Command Category: Node Manager Commands

Use with WLST: Online or Offline

3.10.2.1 Description

Connects WLST to Node Manager to establish a session. After connecting to Node Manager, you can invoke any Node Manager commands via WLST. Node Manager must be running before you can execute this command.

> **Note:** If you have previously used the connect command in the current WLST session, nmconnect uses the same user credentials as were used for the connect command, unless you specify otherwise.

Once connected, the WLST prompt displays as follows, where *domainName* indicates the name of the WebLogic domain that is being managed: wls:/nm/*domainName*>. If you then connect WLST to a WebLogic Server instance, the prompt is changed to reflect the WebLogic Server instance. You can use the nm command to determine whether WLST is connected to Node Manager, as described in Section 3.10.1, "nm".

In the event of an error, the command returns a WLSTException.

3.10.2.2 Syntax

```
nmConnect([username, password], [host], [port], [domainName], [domainDir]
[nmType], [verbose])

nmConnect([userConfigFile, userKeyFile], [host], [port], [domainName], [domainDir]
[nmType], [verbose])
```

Argument	Definition
username	Username of the operator who is connecting WLST to Node Manager. The username defaults to weblogic.
	Note: When running a server in production mode, you must specify the username and password explicitly on the command line to ensure that the appropriate username and password are used when connecting to Node Manager.
password	Password of the operator who is connecting WLST to Node Manager. The password defaults to welcome1.
	Note: When running a server in production mode, you must specify the username and password explicitly on the command line to ensure that the appropriate username and password are used when connecting to Node Manager.
host	Optional. Host name of Node Manager. This argument defaults to localhost.
port	Optional. Port number of Node Manager. This argument defaults to a value that is based on the Node Manager type, as follows:
	■ For plain type, defaults to 5556
	■ For rsh type, defaults to 514
	■ For ssh type, defaults to 22
	■ For ssl type, defaults to 5556
domainName	Optional. Name of the WebLogic domain that you want to manage. This argument defaults to mydomain.
domainDir	Optional. Path of the domain directory to which you want to save the Node Manager secret file (nm_password.properties) and SerializedSystemIni.dat file. This argument defaults to the directory in which WLST was started.

Argument	Definition
nmType	The Node Manager type. Valid values are: ■ `plain` for plain socket Java-based implementation **Note:** If you specify `plain` for `nmType`, you must manually set the `SecureListener` parameter in *WL_HOME*/`common/nodemanager/nodemanager.properties` to `false`. Otherwise, the `nmConnect` command will fail. ■ `rsh` for RSH implementation ■ `ssh` for script-based SSH implementation ■ `ssl` for Java-based SSL implementation This argument defaults to `ssl`.
verbose	Optional. Boolean value specifying whether WLST connects to Node Manager in verbose mode. This argument defaults to `false`, disabling verbose mode.
userConfigFile	Optional. Name and location of a user configuration file which contains an encrypted username and password. When you create a user configuration file, the `storeUserConfig` command uses a key file to encrypt the username and password. Only the key file that encrypts a user configuration file can decrypt the username and password. (See Section 3.8.21, "storeUserConfig".)
userKeyFile	Optional. Name and location of the key file that is associated with the specified user configuration file and is used to decrypt it. (See Section 3.8.21, "storeUserConfig".)

3.10.2.3 Example

The following example connects WLST to Node Manager to monitor the `oamdomain` domain using the default host and port numbers and `plain` Node Manager type.

```
wls:/myserver/serverConfig> nmConnect('weblogic', 'welcome1', 'localhost',
'5555', 'oamdomain', 'c:/Oracle/Middleware/user_projects/domains/oamdomain','ssl')
Connecting to Node Manager Server ...
Successfully connected to Node Manager.
wls:/nm/oamdomain>
```

The following example connects WLST to a Node Manager Server instance using a user configuration and key file to provide user credentials.

```
wls:/myserver/serverConfig> nmConnect(userConfigFile='
c:/myfiles/myuserconfigfile.secure',
userKeyFile='c:/myfiles/myuserkeyfile.secure',
host='172.18.137.82', port=26106, domainName='mydomain',
domainDir='c:/myfiles/mydomain', mType='ssl')
Connecting to Node Manager Server ...
Successfully connected to Node Manager.
wls:/nm/mydomain>
```

3.10.3 nmDisconnect

Command Category: Node Manager Commands

Use with WLST: Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.3.1 Description

Disconnects WLST from a Node Manager session.

In the event of an error, the command returns a `WLSTException`.

3.10.3.2 Syntax

```
nmDisconnect()
```

3.10.3.3 Example

The following example disconnects WLST from a Node Manager session.

```
wls:/nm/oamdomain> nmDisconnect()
Successfully disconnected from Node Manager
wls:/myserver/serverConfig>
```

3.10.4 nmEnroll

Command Category: Node Manager Commands

Use with WLST: Online

3.10.4.1 Description

Enrolls the machine on which WLST is currently running. WLST must be connected to an Administration Server to run this command; WLST does not need to be connected to Node Manager.

This command downloads the following files from the Administration Server:

- Node Manager secret file (`nm_password.properties`), which contains the encrypted username and password that is used for server authentication

- `SerializedSystemIni.dat` file

This command also updates the `nodemanager.domains` file under the *WL_HOME*/`common/nodemanager` directory with the domain information, where *WL_HOME* refers to the top-level installation directory for WebLogic Server.

You must run this command once per WebLogic domain per machine unless that domain shares the root directory of the Administration Server.

If the machine is already enrolled when you run this command, the Node Manager secret file (`nm_password.properties`) is refreshed with the latest information from the Administration Server.

In the event of an error, the command returns a `WLSTException`.

3.10.4.2 Syntax

```
nmEnroll([domainDir], [nmHome])
```

Argument	Definition
domainDir	Optional. Path of the domain directory to which you want to save the Node Manager secret file (`nm_password.properties`) and `SerializedSystemIni.dat` file. This argument defaults to the directory in which WLST was started.
nmHome	Optional. Path to the Node Manager home. The `nodemanager.domains` file, containing the domain information, is written to this directory. This argument defaults to *WL_HOME*/`common/nodemanager`, where *WL_HOME* refers to the top-level installation directory for WebLogic Server.

3.10.4.3 Example

The following example enrolls the current machine with Node Manager and saves the Node Manager secret file (`nm_password properties`) and `SerializedSystemIni.dat` file to `c:/Oracle/Middleware/mydomain/common/nodemanager/nm_password.properties`. The `nodemanager.domains` file is written to *WL_HOME*/`common/nodemanager` by default.

```
wls:/mydomain/serverConfig>
nmEnroll('c:/Oracle/Middleware/mydomain/common/nodemanager')
Enrolling this machine with the domain directory at
c:\Oracle\Middleware\mydomain\common\nodemanager....
Successfully enrolled this machine with the domain directory at
C:\Oracle\Middleware\mydomain\common\nodemanager
wls:/mydomain/serverConfig>
```

3.10.5 nmGenBootStartupProps

Command Category: Node Manager Commands

Use with WLST: Online

3.10.5.1 Description

Generates the Node Manager property files, `boot.properties` and `startup.properties`, for the specified server. The Node Manager property files are stored relative to the root directory of the specified server. The target root directory must be on the same machine on which you are running the command.

You must specify the name of a server; otherwise, the command will fail.

In the event of an error, the command returns a `WLSTException`.

3.10.5.2 Syntax

```
nmGenBootStartupProps(serverName)
```

Argument	Definition
serverName	Name of the server for which Node Manager property files are generated.

3.10.5.3 Example

The following example generates `boot.properties` and `startup.properties` in the root directory of the specified server, `ms1`.

```
wls:/mydomain/serverConfig> nmGenBootStartupProps('ms1')
Successfully generated boot.properties at
c:\Oracle\Middleware\mydomain\servers\ms1\data\nodemanager\boot.properties
Successfully generated startup.properties at
c:\Oracle\Middleware\mydomain\servers\ms1\data\nodemanager\startup.properties
wls:/mydomain/serverConfig>
```

3.10.6 nmKill

Command Category: Node Manager Commands

Use with WLST: Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.6.1 Description

Kills the specified server instance that was started with Node Manager.

If you do not specify a server name using the *serverName* argument, the argument defaults to mySerer, which must match your server name or the command will fail.

If you attempt to kill a server instance that was not started using Node Manager, the command displays an error.

In the event of an error, the command returns a WLSTException.

3.10.6.2 Syntax

nmKill([serverName], [serverType])

Argument	Definition
serverName	Optional. Name of the server to be killed. This argument defaults to myserver.
serverType	Optional. The type of server to start. This argument defaults to WebLogic. Another valid option is Coherence.

3.10.6.3 Example

The following example kills the server named oamserver.

```
wls:/nm/oamdomain> nmKill('oamserver')
Killing server 'oamserver' ...
Server oamServer killed successfully.
wls:/nm/oamdomain>
```

3.10.7 nmLog

Command Category: Node Manager Commands

Use with WLST: Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.7.1 Description

Returns the Node Manager log.

In the event of an error, the command returns a WLSTException.

3.10.7.2 Syntax

nmLog([writer])

Argument	Definition
writer	Optional. java.io.Writer object to which you want to stream the log output. This argument defaults to the WLST writer stream.

3.10.7.3 Example

The following example displays the Node Manager log.

```
wls:/nm/oamdomain> nmLog()
Successfully retrieved the Node Manager log and written.
wls:/nm/oamdomain>
```

3.10.8 nmServerLog

Command Category: Node Manager Commands

Use with WLST: Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.8.1 Description

Returns the server output log of the server that was started with Node Manager.

In the event of an error, the command returns a WLSTException.

3.10.8.2 Syntax

nmServerLog([serverName], [writer], [serverType])

Argument	Definition
serverName	Optional. Name of the server for which you want to display the server output log. This argument defaults to myserver.
writer	Optional. java.io.Writer object to which you want to stream the log output. This argument defaults to the WLSTInterpreter standard out, if not specified.
serverType	Optional. The type of server to start. This argument defaults to WebLogic. Another valid option is Coherence.

3.10.8.3 Example

The following example displays the server output log for the oamserver server and writes the log output to myWriter.

```
wls:/nm/oamdomain> nmServerLog('oamserver',myWriter)
Successfully retrieved the server log and written.
wls:/nm/oamdomain>
```

3.10.9 nmServerStatus

Command Category: Node Manager Commands

Use with WLST: Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.9.1 Description

Returns the status of the server that was started with Node Manager.

In the event of an error, the command returns a WLSTException.

3.10.9.2 Syntax

nmServerStatus([serverName], [serverType])

Argument	Definition
serverName	Optional. Name of the server for which you want to display the status. This argument defaults to myserver.
serverType	Optional. The type of server to start. This argument defaults to WebLogic. Another valid option is Coherence.

3.10.9.3 Example

The following example displays the status of the server named oamserver, which was started with Node Manager.

```
wls:/nm/oamdomain> nmServerStatus('oamserver')
RUNNING
wls:/nm/oamdomain>
```

3.10.10 nmStart

Command Category: Node Manager Commands

Use with WLST: Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.10.1 Description

Starts a server in the current WebLogic domain using Node Manager.

In the event of an error, the command returns a WLSTException.

> **Note:** boot.properties must exist in order to start a server with nmStart. If this is the first time you are starting a server, you must manually create it in order to use nmStart.
>
> Alternatively, you can use the nmStartprops argument to provide user credentials (after connecting to Node Manager):
>
> ```
> prps = makePropertiesObject("username=weblogic, password=welcome1")
> nmStart("AdminServer",props=prps)
> ```

3.10.10.2 Syntax

nmStart([serverName], [domainDir], [props], [writer], [serverType])

Argument	Definition
serverName	Optional. Name of the server to be started.
domainDir	Optional. Domain directory of the server to be started. This argument defaults to the directory from which you started WLST.
props	Optional. System properties to apply to the new server.
writer	Optional. java.io.Writer object to which the server output is written. This argument defaults to the WLST writer.
serverType	Optional. The type of server to start. This argument defaults to WebLogic. Another valid option is Coherence.

3.10.10.3 Example

The following example starts the managed1 server in the current WebLogic domain using Node Manager.

```
wls:/nm/mydomain> nmStart("managed1")
Starting server managed1 ...
Server managed1 started successfully
wls:/nm/mydomain>
```

The following example starts the Administration Server in the specified WebLogic domain using Node Manager. In this example, the prps variable stores the system property settings and is passed to the command using the props argument.

```
wls:/nm/mydomain> prps = makePropertiesObject("weblogic.ListenPort=8001")
wls:/nm/mydomain> nmStart("AdminServer",props=prps)
Starting server AdminServer...
Server AdminServer started successfully
wls:/nm/mydomain>
```

3.10.11 nmVersion

Command Category: Node Manager Commands

Use with WLST: Online or Offline

WLST must be connected to Node Manager to run this command.

3.10.11.1 Description
Returns the Node Manager version.

In the event of an error, the command returns a WLSTException.

3.10.11.2 Syntax
```
nmVersion()
```

3.10.11.3 Example
The following example displays the Node Manager version.

```
wls:/nm/oamdomain> nmVersion()
The Node Manager version that you are currently connected to is 9.0.0.0
wls:/nm/oamdomain>
```

3.10.12 startNodeManager

Command Category: Node Manager Commands

Use with WLST: Online or Offline

3.10.12.1 Description
Starts Node Manager on the same computer that is running WLST.

Notes: The WebLogic Server custom installation process optionally installs and starts Node Manager as a Windows service on Windows systems. For more information, see "About Installing Node Manager as a Windows Service" in the *Installation Guide for Oracle WebLogic Server*. In this case, you do not need to start the Node Manager manually.

In production environments, Oracle recommends that you do *not* use the startNodeManager command to start Node Manager. The recommended approach is to install Node Manager as a service or daemon, or to use the startNodeManager script (startNodeManager.sh or startNodeManger.cmd).

If Node Manager is already running when you invoke the `startNodeManager` command, the following message is displayed:

```
A Node Manager has already been started.
Cannot start another Node Manager process via WLST
```

In the event of an error, the command returns a `WLSTException`.

3.10.12.2 Syntax

```
startNodeManager([verbose], [nmProperties])
```

Argument	Definition
verbose	Optional. Boolean value specifying whether WLST starts Node Manager in verbose mode. This argument defaults to `false`, disabling verbose mode.
nmProperties	Optional. Comma-separated list of Node Manager properties, specified as name-value pairs. Node Manager properties include, but are not limited to, the following: `NodeManagerHome`, `ListenAddress`, `ListenPort`, and `PropertiesFile`.

3.10.12.3 Example

The following example displays the Node Manager server version.

```
wls:/mydomain/serverConfig> startNodeManager(verbose='true',
NodeManagerHome='c:/Oracle/Middleware/wlserver_12.1/common/nodemanager',
ListenPort='6666', ListenAddress='myhost'))
Launching Node Manager ...
Successfully launched the Node Manager.
The Node Manager process is running independent of the WLST process
Exiting WLST will not stop the Node Manager process. Please refer
to the Node Manager logs for more information.
The Node Manager logs will be under c:\Oracle\Middleware\wlserver_
12.1\common\nodemanager.
wls:/mydomain/serverConfig>
```

3.10.13 stopNodeManager

Command Category: Node Manager Commands

Use with WLST: Online or Offline

3.10.13.1 Description

Stops the Node Manager process.

> **Note:** In order to stop the Node Manager process, you must have
> either started Node Manager with `startNodeManager`, or Node
> Manager must have been started with the property
> `QuitEnabled=true`. You can configure this property in `$WLS_`
> `HOME/common/nodemanager.properties`. This allows you to
> connect to the Node Manager to shut it down.

If the Node Manager is not running when you invoke the `stopNodeManager` command, the following message is displayed:

```
Cannot stop the Node Manager unless you are connected to it.
```

3.10.13.2 Syntax

```
stopNodeManager()
```

3.10.13.3 Example

The following example stops the Node Manager process for the base_domain domain.

```
wls:/nm/base_domain> stopNodeManager()
Stopped Node Manager Process successfully
wls:/offline>
```

3.11 Tree Commands

Use the WLST tree commands, listed in Table 3–12, to navigate among MBean hierarchies.

Table 3–12   Tree Commands for WLST Configuration

Use this command...	To...	Use with WLST...
custom	Navigate to the root of custom MBeans that are registered in the server.	Online
domainConfig	Navigate to the last MBean to which you navigated in the domain configuration hierarchy or to the root of the hierarchy, DomainMBean.	Online
domainCustom	Navigate to the root of custom MBeans that are registered in the Domain Runtime MBean Server	Online
domainRuntime	Navigate to the last MBean to which you navigated in the domain runtime hierarchy or to the root of the hierarchy, DomainRuntimeMBean.	Online
edit	Navigate to the last MBean to which you navigated in the edit configuration MBean hierarchy or to the root of the hierarchy, DomainMBean.	Online
jndi	Navigates to the JNDI tree for the server to which WLST is currently connected.	Online
serverConfig	Navigate to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, DomainMBean.	Online
serverRuntime	Navigate to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, ServerRuntimeMBean.	Online

3.11.1 custom

Command Category: Tree Commands

Use with WLST: Online

3.11.1.1 Description

Navigates to the root of custom MBeans that are registered in the Runtime MBean Server. WLST navigates, interrogates, and edits custom MBeans as it does domain MBeans; however, custom MBeans cannot use the cmo variable because a stub is not available.

> **Note:** When navigating to the custom tree, WLST queries all MBeans in the compatibility MBean server, the runtime MBean server, and potentially the JVM platform MBean server to locate the custom MBeans. Depending on the number of MBeans in the current WebLogic domain, this process make take a few minutes, and WLST may not return a prompt right away.

The custom command is available when WLST is connected to an Administration Server instance or a Managed Server instance. When connected to a WebLogic Integration or WebLogic Portal server, WLST can interact with all the WebLogic Integration or WebLogic Portal server MBeans.

For more information about custom MBeans, see *Developing Custom Management Utilities With JMX for Oracle WebLogic Server*.

In the event of an error, the command returns a WLSTException.

> **Note:** You can also navigate to custom MBeans on the Domain Runtime MBean Server using the domainCustom() command. See Section 3.11.3, "domainCustom," for more information.

3.11.1.2 Syntax

```
custom()
```

3.11.1.3 Example

The following example navigates from the configuration MBean hierarchy to the custom MBean hierarchy on a Administration Server instance.

```
wls:/mydomain/serverConfig> custom()
Location changed to custom tree. This is a writeable tree with No root. For more
help, use help('custom')
wls:/mydomain/custom>
```

3.11.2 domainConfig

Command Category: Tree Commands

Use with WLST: Online

3.11.2.1 Description

Navigates to the last MBean to which you navigated in the domain Configuration hierarchy or to the root of the hierarchy, DomainMBean. This read-only hierarchy stores the configuration MBeans that represent your current WebLogic domain.

In the event of an error, the command returns a WLSTException.

3.11.2.2 Syntax

```
domainConfig()
```

3.11.2.3 Example

The following example navigates from the configuration MBean hierarchy to the WebLogic domain Configuration hierarchy on an Administration Server instance.

```
wls:/mydomain/serverConfig> domainConfig()
```

```
Location changed to domainConfig tree. This is a read-only tree with DomainMBean
as the root.
For more help, use help('domainConfig')
wls:/mydomain/domainConfig> ls()
dr--   AppDeployments
dr--   BridgeDestinations
dr--   Clusters
dr--   CustomResources
dr--   DeploymentConfiguration
dr--   Deployments
dr--   EmbeddedLDAP
dr--   ErrorHandlings
dr--   FileStores
dr--   InternalAppDeployments
dr--   InternalLibraries
dr--   JDBCDataSourceFactories
dr--   JDBCStores
dr--   JDBCSystemResources
dr--   JMSBridgeDestinations
dr--   JMSInteropModules
dr--   JMSServers
dr--   JMSSystemResources
...
wls:/mydomain/domainConfig>
```

3.11.3 domainCustom

Command Category: Tree Commands

Use with WLST: Online

3.11.3.1 Description

Navigates to the domain custom tree of custom MBeans that are registered in the
Domain Runtime MBean Server. WLST navigates, interrogates, and edits domain
custom MBeans as it does domain MBeans; however, domain custom MBeans cannot
use the cmo variable because a stub is not available.

Note: When navigating to the domainCustom tree, WLST queries all
MBeans in the Domain Runtime MBean Server, the Runtime MBean
Servers on each server, and potentially the JVM platform MBean
server to locate the custom MBeans. Depending on the number of
MBeans in the current WebLogic domain, this process make take a few
minutes, and WLST may not return a prompt right away. It is
recommended that a JMX query Object Name Pattern be specified to
limit the amount of searching performed.

The domainCustom command is available only when WLST is connected to an
Administration Server instance.

For more information about the Domain Runtime MBean Server, see "Understanding
WebLogic Server MBeans" in *Developing Custom Management Utilities With JMX for
Oracle WebLogic Server*.

In the event of an error, the command returns a WLSTException.

3.11.3.2 Syntax

```
domainCustom(ObjectNamePattern)
```

Argument	Definition
ObjectNamePattern	A JMX query pattern, such as sip:*. The default value is null or *:*.

3.11.3.3 Example

The following example navigates from the configuration MBean hierarchy to the domain custom MBean hierarchy on an Administration Server instance:

```
wls:/mydomain/serverConfig> domainCustom()
Location changed to domain custom tree. This is a writeable tree with No root. For
more help, use help('domainCustom').

wls:/mydomain/domainCustom
```

3.11.4 domainRuntime

Command Category: Tree Commands

Use with WLST: Online

3.11.4.1 Description

Navigates to the last MBean to which you navigated in the domain Runtime hierarchy or to the root of the hierarchy, DomainRuntimeMBean. This read-only hierarchy stores the runtime MBeans that represent your current WebLogic domain.

In the event of an error, the command returns a WLSTException.

3.11.4.2 Syntax

```
domainRuntime()
```

3.11.4.3 Example

The following example navigates from the configuration MBean hierarchy to the domain Runtime hierarchy on an Administration Server instance.

```
wls:/mydomain/serverConfig> domainRuntime()
wls:/mydomain/domainRuntime> ls()
dr--   AppRuntimeStateRuntime
dr--   DeployerRuntime
dr--   DomainServices
dr--   LogRuntime
dr--   MessageDrivenControlEJBRuntime
dr--   MigratableServiceCoordinatorRuntime
dr--   MigrationDataRuntimes
dr--   SNMPAgentRuntime
dr--   ServerLifeCycleRuntimes
dr--   ServerRuntimes
dr--   ServerServices

-r--   ActivationTime                        Mon Aug 01 11:41:25 EDT 2005
-r--   Clusters                              null
-r--   MigrationDataRuntimes                 null
-r--   Name                                  sampleMedRecDomain
-rw-   Parent                                null
-r--   SNMPAgentRuntime                      null
```

```
-r--    Type                                        DomainRuntime
-r-x    restartSystemResource                       Void :
        WebLogicMBean(weblogic.management.configuration.SystemResourceMBean)
wls:/mydomain/domainRuntime>
```

3.11.5 edit

Command Category: Tree Commands

Use with WLST: Online

3.11.5.1 Description

Navigates to the last MBean to which you navigated in the edit configuration MBean hierarchy or to the root of the hierarchy, `DomainMBean`. This writable hierarchy stores all of the configuration MBeans that represent your current WebLogic domain.

> **Note:** To edit configuration beans, you must be connected to an Administration Server. If you connect to a Managed Server, WLST functionality is limited to browsing the configuration bean hierarchy. While you cannot use WLST to change the values of MBeans on Managed Servers, it is possible to use the Management APIs to do so. Oracle recommends that you change only the values of configuration MBeans on the Administration Server. Changing the values of MBeans on Managed Servers can lead to an inconsistent domain configuration.
>
> For more information about editing configuration beans, see "Using WLST Online to Update an Existing Domain" in *Oracle WebLogic Scripting Tool*.

In the event of an error, the command returns a `WLSTException`.

3.11.5.2 Syntax

```
edit()
```

3.11.5.3 Example

The following example illustrates how to navigate from the server configuration MBean hierarchy to the editable copy of the domain configuration MBean hierarchy, in an Administration Server instance.

```
wls:/myserver/serverConfig> edit()
Location changed to edit tree. This is a writeable tree with DomainMBean as the
root.
For more help, use help('edit')
wls:/myserver/edit !> ls()
dr--    AppDeployments
dr--    BridgeDestinations
dr--    Clusters
dr--    DeploymentConfiguration
dr--    Deployments
dr--    EmbeddedLDAP
...
wls:/myserver/edit !>
```

3.11.6 jndi

Command Category: Tree Commands

Use with WLST: Online

3.11.6.1 Description

Navigates to the JNDI tree for the server to which WLST is currently connected. This read-only tree holds all the elements that are currently bound in JNDI.

In the event of an error, the command returns a WLSTException.

3.11.6.2 Syntax

```
jndi()
```

3.11.6.3 Example

The following example navigates from the runtime MBean hierarchy to the Domain JNDI tree on an Administration Server instance.

```
wls:/myserver/runtime> jndi()
Location changed to jndi tree. This is a read-only tree with No root. For more
help, use help('jndi')
wls:/myserver/jndi> ls()
dr--   ejb
dr--   javax
dr--   jms
dr--   weblogic
...
```

3.11.7 serverConfig

Command Category: Tree Commands

Use with WLST: Online

3.11.7.1 Description

Navigates to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, `DomainMBean`.

This read-only hierarchy stores the configuration MBeans that represent the server to which WLST is currently connected. The MBean attribute values include any command-line overrides that a user specified while starting the server.

In the event of an error, the command returns a `WLSTException`.

For more information, see "Navigating Among MBean Hierarchies" in *Oracle WebLogic Scripting Tool*.

3.11.7.2 Syntax

```
serverConfig()
```

3.11.7.3 Example

The following example navigates from the domain runtime MBean hierarchy to the configuration MBean hierarchy on an Administration Server instance.

```
wls:/mydomain/domainRuntime> serverConfig()
wls:/mydomain/serverConfig>
```

3.11.8 serverRuntime

Command Category: Tree Commands

Use with WLST: Online

3.11.8.1 Description

Navigates to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, `ServerRuntimeMBean`. This read-only hierarchy stores the runtime MBeans that represent the server to which WLST is currently connected.

In the event of an error, the command returns a `WLSTException`.

3.11.8.2 Syntax

```
serverRuntime()
```

3.11.8.3 Example

The following example navigates from the configuration MBean hierarchy to the runtime MBean hierarchy on an Administration Server instance.

```
wls:/mydomain/serverConfig> serverRuntime()
Location changed to serverRuntime tree. This is a read-only tree with
ServerRuntimeMBean as the root.
For more help, use help('serverRuntime')
wls:/mydomain/serverRuntime>
```

3.12 WLST Variable Reference

Table 3–13 describes WLST variables and their common usage. All variables are initialized to default values at the start of a user session and are changed according to the user interaction with WLST.

Table 3–13   WLST Variables

Variable	Description	Example
cmgr	The cmgr variable is set to the ConfigurationManagerMBean. You can use this variable to get the current value of any ConfigurationManagerMBean attribute.	wls:/mydomain/edit> **cmgr.getCurrentEditor()** 'weblogic'
cmo	Current Management Object. The cmo variable is set to the bean instance to which you navigate using WLST. You can use this variable to perform any get, set, or invoke method on the current bean instance.	wls:/mydomain/edit> **cmo.setAdministrationPort(9092)**
	WLST sets the variable to the current WLST path. For example, when you change to the serverConfig hierarchy, cmo is set to DomainMBean. When you change to the serverRuntime hierarchy, cmo is set to ServerRuntimeMBean.	
	The variable is available in all WLST hierarchies except custom and jndi.	
connected	Boolean value specifying whether WLST is connected to a running server. WLST sets this variable to true when connected to a running server; otherwise, WLST sets it to false.	wls:/mydomain/serverConfig> **print connected** false
domainName	Name of the WebLogic domain to which WLST is connected.	wls:/mydomain/serverConfig> **print domainName** mydomain
domainRuntimeService	DomainRuntimeServiceMBean MBean. This variable is available only when WLST is connected to the Administration Server.	wls:/mydomain/serverConfig> **domainService.getServerName()** 'myserver'
editService	EditServiceMBean MBean. This variable is available only when WLST is connected to the Administration Server.	wls:/mydomain/edit> **dc = editService.getDomainConfiguration()**
exitonerror	Boolean value specifying whether WLST terminates script execution when it encounters an exception. This variable defaults to true, indicating that script execution is terminated when WLST encounters an error. This variable is not applicable when running WLST in interactive mode.	wls:/mydomain/serverConfig> **print exitonerror** true
home	Represents the local MBeanHome.	wls:/mydomain/serverConfig> **print home** weblogic.rmi.internal.BasicRemoteRef - hostID: '-*hostID*:[7001,7001,-1,-1,-1,-1,-1]:mydomain:Admin Server', oid: '260', channel: 'null'
isAdminServer	Boolean value specifying whether WLST is connected to a WebLogic Administration Server instance. WLST sets this variable to true if WLST is connected to a WebLogic Administration Server; otherwise, WLST sets it to false.	wls:/mydomain/serverConfig> **print isAdminServer** true

Table 3–13 (Cont.) WLST Variables

Variable	Description	Example
mbs	MBeanServerConnection object that corresponds to the current location in the hierarchy.	wls:/mydomain/serverConfig> **mbs.isRegistered(ObjectName('mydomain: Name=mydomain,Type=Domain'))**
recording	Boolean value specifying whether WLST is recording commands. WLST sets this variable to true when the startRecording command is entered; otherwise, WLST sets this variable to false.	wls:/mydomain/serverConfig> **print recording** true
runtimeService	RuntimeServiceMBean MBean.	wls:/mydomain/serverConfig> **sr=runtimeService.getServerRuntime()**
serverName	Name of the server to which WLST is connected.	wls:/mydomain/serverConfig> **print serverName** myserver
typeService	TypeServiceMBean MBean.	wls:/mydomain/serverConfig> **mi=typeService.getMBeanInfo('weblogic. management.configuration.ServerMBean')**
username	Name of user currently connected to WLST.	wls:/mydomain/serverConfig> **print username** weblogic
version	Current version of the running server to which WLST is connected.	wls:/mydomain/serverConfig> **print version** WebLogic Server 9.0 Thu Aug 31 12:15:50 PST 2005 778899