Oracle® Fusion Middleware

WebLogic Scripting Tool Command Reference

11*g* Release 1 (10.3.1)

E13813-01

May 2009

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, 11*g* Release 1 (10.3.1)

E13813-01

Contents

4 Infrastructure Security Custom WLST Commands

5 Oracle WebCenter Custom WLST Commands

10 Oracle SOA Suite Custom WLST Commands

11 Application Development Framework (ADF) Custom WLST Commands

12 Portal Custom WLST Commands

13 Java Required Files Custom WLST Commands

Preface

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

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at http://www.fcc.gov/cgb/consumerfacts/trs.html, and a list of phone numbers is available at http://www.fcc.gov/cgb/dro/trsphonebk.html.

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 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 availabe 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 Sun Microsystems. 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.
- Chapter 4, "Infrastructure Security Custom WLST Commands," provides detailed descriptions for each of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Infrastructure Security components.

- Chapter 5, "Oracle WebCenter Custom WLST Commands," provides detailed descriptions for each of the custom WLST commands that can be used to manage the Oracle Fusion Middleware WebCenter component.
- Chapter 6, "User Messaging Service (UMS) Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware User Messaging Service (UMS) component.
- Chapter 7, "DMS Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Dynamic Monitoring Service (DMS) component.
- Chapter 8, "Logging Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Logging component.
- Chapter 9, "Metadata Services (MDS) Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Metadata Services (MDS) component.
- Chapter 10, "Oracle SOA Suite Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware SOA component.
- Chapter 11, "Application Development Framework (ADF) Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware ADF component.
- Chapter 12, "Portal Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Portals component.
- Chapter 13, "Java Required Files Custom WLST Commands,"provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware JRF component.

1.3 Related Documentation

For information about how to use the WebLogic Scripting Tool, refer to *Oracle Fusion Middleware 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 *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*, describes using WebLogic Ant tasks for starting and stopping WebLogic Server instances and configuring WebLogic Server domains.
- "Deployment Tools" in *Oracle Fusion Middleware 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 Server domains.
- *Creating WebLogic Domains Using the Configuration Wizard* describes using a graphical user interface to create a WebLogic Server domain or extend an existing one.
- *Creating Templates and Domains Using the Pack and Unpack Commands* describes commands that recreate existing domains quickly and easily.

- *Oracle Fusion Middleware Developing Custom Management Utilities With JMX for Oracle WebLogic Server* describes using Java Management Extensions (JMX) APIs to monitor and modify WebLogic Server resources.
- *Oracle Fusion Middleware SNMP Management Guide for Oracle WebLogic Server* describes using Simple Network Management Protocol (SNMP) to monitor WebLogic Server domains.

1.4 New and Changed WLST Features in This Release

The following new WLST features are available in this release:

- Support for using WLST with some Oracle Fusion Middleware products. Custom WLST commands for these products are described in this manual. These commands are WLST versions of many of the Oracle ASCTL commands for Fusion Middleware products.
- A new `domainCustom()` command that lets you access and perform operations on custom MBeans that have been registered in the Domain Runtime MBean Server. See "Accessing Custom MBeans on the Domain Runtime Server" in *Oracle Fusion Middleware Oracle WebLogic Scripting Tool*, and Section 3.11.3, "domainCustom" in this manual.
- Two new WLST commands, `addHelpCommandGroup()` and `addHelpCommand()`, which lets you add command group help and command help for custom WLST commands to the WLST integrated help. See Section 3.4.1, "addHelpCommandGroup" and Section 3.4.2, "addHelpCommand".
- As of this release, WLST is based on Jython 2.2.1.

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

2

WebLogic Server WLST Online and Offline Command Reference

The following sections summarize the WebLogic Server WLST commands, as follows:

- 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")
```

For information about custom WLST commands for Fusion Middleware (FMW) components, refer to the appropriate chapter in this document. For information on how to run FMW custom commands, see "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide*.

2.1 WebLogic Server WLST Command Summary, Alphabetically By Command

The following tables summarizes each of the WebLogic Server WLST commands, alphabetically by command. This table does not include custom WLST commands for FMW components. For a list of custom commands for a given FMW component, refer to the appropriate chapter in this document.

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

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

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
addListener	Add a JMX listener to the specified MBean.	Online
addTemplate	Extend the current domain using an application or service extension template.	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
cd	Navigate the hierarchy of configuration or runtime beans.	Online or Offline
closeDomain	Close the current 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

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

This command...	Enables you to...	Use with WLST...
exit	Exit WLST from the user session and close the scripting shell.	Online or Offline
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
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 domain for updating.	Offline
readTemplate	Open an existing domain template for 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
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
set	Set the specified attribute value for the current configuration bean.	Online or Offline
setOption	Set options related to a domain creation or update	Offline
showChanges	Show the changes made by the current user during the current edit session.	Online

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

This command...	Enables you to...	Use with WLST...
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
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
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
readTemplate	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

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

This command...	Enables you to...	Use with WLST...
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. This table does not include custom WLST commands for FMW components. For a list of custom commands for a given FMW component, refer to the appropriate chapter in this document.

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.
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.

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

This command...	Enables you to...
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.
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.
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.

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

This command...	Enables you to...
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.
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.

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

This command...	Enables you to...
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.
stopRecording	Stop recording WLST commands.
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.

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

This command...	Enables you to...
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.
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 domain for updating.
readTemplate	Open an existing domain template for domain creation.
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 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.
stopRecording	Stop recording WLST commands.
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.
readTemplate	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

The following sections describe the WLST commands and variables in detail. Topics include:

3.1 Overview of WSLT Command Categories

Note: It is recommended that you review "Syntax for WLST Commands" in *Oracle Fusion Middleware 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 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 3–13, 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 Fusion Middleware 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 3–13, to perform the following tasks:

- Connect to or disconnect from a server
- Create and configure a WebLogic domain or domain template, similar to the Configuration Wizard
- Exit WLST

Table 3–13 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

Table 3–3 (Cont.) Control Commands for WLST Configuration

Use this command...	To...	Use with WLST...
createDomain	Create a new 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 domain for updating.	Offline
addTemplate	Extend the current 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 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 domain and extends it using the specified extension template, `DefaultWebApp.jar`.

```
wls:/offline> readDomain('c:/bea/user_projects/domains/wlw')
wls:/offline/wlw> addTemplate('c:/bea/wlserver_10.3/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:/bea/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:/bea/wlserver_10.3/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, WebLogic Server encrypts the credentials used the create the server and stores them in a `boot.properties` file.

If you run the `connect` command without specifying the username and password or user configuration file and key file, WLST attempts to process the command using one of the methods listed below (in order of precedence):

1. If a user configuration and default key file exists in your home directory, then use those files. The location of the home directory depends on the type of operating system on which WLST is running. For information about the default location, see Section 3.8.21, "storeUserConfig".
2. If the `adminServerName` argument is not specified, then look for the `boot.properties` file in `./boot.properties` or `./servers/myserver/security/boot.properties`.
3. If the `adminServerName` argument is specified, then look for the `boot.properties` file in `./servers/`*`adminServerName`*`/security/boot.properties`, where *`adminServerName`* is the value of the `adminServerName` argument.

Please note:

- 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 Fusion Middleware 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 Fusion Middleware Oracle WebLogic Server MBean Reference*.

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`.
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".

Argument	Definition
adminServerName	Optional. Name of the domain's Administration Server. 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 Administration Server's `boot.properties` file 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 weblogic server instance running at t3://localhost:7001 as
username ...

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

3.3.5 createDomain

Command Category: Control Commands

Use with WLST: Offline

3.3.5.1 Description

Creates a domain using the specified template.

> **Note:** If you wish to modify the domain configuration settings when creating a domain, see Option 2 in "Editing a Domain (Offline)" in *Oracle Fusion Middleware 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.
user	Name of the default user.
password	Password of the default user.

3.3.5.3 Example

The following example creates a new 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:/bea/user_projects/domains/medrec`.

```
wls:/offline> createDomain('c:/bea/wlserver_10.3/common/templates/domains
/wls_medrec.jar','c:/bea/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 domain for updating.

WLST offline provides read and write access to the configuration data that is persisted in the domain's `config` directory 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 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 Fusion Middleware 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 domain directory that you wish to open.

3.3.8.3 Example

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

```
wls:/offline> readDomain('c:/bea/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 Fusion Middleware 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 domain. When you are updating a 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 domain creation.

```
wls:/offline> readTemplate('c:/bea/wlserver_10.3/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 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:/bea/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 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 domain, the associated applications are written to *BEAHOME*`/user_projects/applications/`*domainname*, where *BEAHOME* specifies the BEA home directory and *domainname* specifies the name of the 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 domain that has been saved to disk, you can open it using the `readDomain` command.

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

```
wls:/offline> readTemplate('c:/bea/wlserver_10.3/common/templates/domains
/wls.jar')
...
wls:/offline/base_domain> writeDomain('c:/bea/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 domain.

Once your write the configuration information to the domain configuration template, you can continue to update the 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 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:/bea/user_projects/templates/myTemplate.jar`.

```
wls:/offline> readDomain('c:/bea/user_projects/domains/mydomain')
...
wls:/offline/base_domain> writeTemplate('c:/bea/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 Fusion Middleware 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

Table 3–4 (Cont.) Customization Commands for WLST Configuration

This command...	Enables you to...	Use with WLST...
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 Fusion Middleware 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 3–13, 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 *Oracle Fusion Middleware 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

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

This command...	Enables you to...	Use with WLST...
getWLDM	Return the WebLogic `DeploymentManager` object.	Online
listApplications	List all applications that are currently deployed in the domain.	Online
loadApplication	Load an application and deployment plan into memory.	Online
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
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 Fusion Middleware 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.

Argument	Definition
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 *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*. This argument defaults to null.
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. 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 Fusion Middleware 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 *Oracle Fusion Middleware 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`. ■ **testMode**—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 Fusion Middleware 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 *Oracle Fusion Middleware 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 Fusion Middleware 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.
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 Fusion Middleware 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 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

3.5.5.1 Description

Loads an application and deployment plan into memory.

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 Fusion Middleware 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:/myserver/serverConfig> 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.
wls:/myserver/serverConfig>
```

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

For more information about the `WLSTPlan` object, see "WLSTPlan Object" in *Oracle Fusion Middleware 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 Fusion Middleware 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 *Oracle Fusion Middleware 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.
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 Fusion Middleware 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 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 Fusion Middleware 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:/offline> progress=startApplication('BigApp', stageMode='NOSTAGE',
testMode='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 Fusion Middleware 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 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 Fusion Middleware 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 Fusion Middleware 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 Fusion Middleware 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 *Oracle Fusion Middleware 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 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 Fusion Middleware 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', testMode='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 Fusion Middleware Oracle WebLogic Scripting Tool*.

3.6 Diagnostics Commands

Use the WLST diagnostics commands, listed in Table 3–13, to retrieve diagnostics data by executing queries against the WebLogic Diagnostics Framework (WLDF) data

stores. For more information about WLDF, see *Oracle Fusion Middleware 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

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 *Oracle Fusion Middleware 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 *Oracle Fusion Middleware 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:\bea\wlserver_
10.3\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 *Oracle Fusion Middleware 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")
```

3.7 Editing Commands

Use the WLST editing commands, listed in Table 3–13, 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 Fusion Middleware 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
isRestartRequired	Determine whether a server restart is required.	Online

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

This command...	Enables you to...	Use with WLST...
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 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 Fusion Middleware 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.
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 Managementt Object" in *Oracle Fusion Middleware 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 Fusion Middleware Oracle WebLogic Scripting Tool*.
- MBeans, their child types, attributes, and operations, see *Oracle Fusion Middleware 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.
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 Fusion Middleware Oracle WebLogic Server MBean Reference*.

Argument	Definition
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')
```

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 Fusion Middleware 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 Fusion Middleware 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 domain in which you want to use the encrypted string. The string can be used only in the 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 domain directory. The encrypted string can be used only by the domain that is contained within the specified directory. If you do not specify this argument, the command encrypts the string for use in the 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 domain directory.

```
wls:/mydomain/serverConfig> es=encrypt('myPassword','c:/bea/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 Fusion Middleware 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 PointBase version 4.4, the SQL file list is located at `_jdbc_\Pointbase\44\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 Fusion Middleware 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's configuration files. When using WLST online, this command sets the value of an MBean attribute. Online changes are written to the domain's 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 Fusion Middleware 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.set`*`attrName`*`(`*`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 Fusion Middleware 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.

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 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 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 domain. By default, the name of the domain is derived from the name of the domain directory. For example, for a domain saved to c:/bea/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 domain to be overwritten. This option defaults to false. ■ ServerStartMode—Mode to use when starting the server for the newly created 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 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 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 *BEAHOME*/user_projects/applications/*domainname*, where *BEAHOME* specifies the BEA home directory and *domainname* specifies the name of the 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 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 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 3–13, 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:\bea\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:\bea\user_projects\domains\wls\config\config.py
and the properties file associated with this script is written to c:\bea\user_
projects\domains\wls\config\config.py.properties
wls:/offline>
```

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

```
wls:/offline> configToScript('c:/bea/user_projects/domains/mydomain',
'c:/bea/myscripts')
configToScript is loading configuration from c:\bea\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:\bea\myscripts\config.py
and the properties file associated with this script is written to
c:\bea\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 1080801509042639375: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 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 within the task 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 Fusion Middleware 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 Fusion Middleware 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 Fusion Middleware 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 Fusion Middleware 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 the following when using WLST offline:

- As a performance optimization, WebLogic Server does not store most of its default values in the domain's configuration files. 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's configuration files). For example, if you never modify the default logging severity level for a domain while the domain is active, WLST offline will not display the domain's `Log` management object.

  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.
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.

Argument	Definition
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.

Argument	Definition
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 Fusion Middleware 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 *Oracle Fusion Middleware 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 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 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 Fusion Middleware 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 3–13, 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 *Oracle Fusion Middleware 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 *Oracle Fusion Middleware 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.
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.

Argument	Definition
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 *Oracle Fusion Middleware 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 Fusion Middleware 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.
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.

Argument	Definition
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 Fusion Middleware 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 Fusion Middleware 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`.

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 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 `weblogic`.
domainDir	Optional. Domain directory in which the Administration Server is being started. This argument defaults to the directory from which you started WLST.
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 Fusion Middleware 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-seperated 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.
spaceAsJvmArgsDelimiter	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 *Oracle Fusion Middleware 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.
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 Fusion Middleware 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 3–13, to start, shut down, restart, and monitor WebLogic Server instances.

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 *Oracle Fusion Middleware 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 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
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 version.	Online or Offline
startNodeManager	Starts Node Manager on the same computer that is running WLST.	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.

Once connected, the WLST prompt displays as follows, where *`domainName`* indicates the name of the 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.

Argument	Definition
password	Password of the operator who is connecting WLST to Node Manager. The password 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.
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 server 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 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.
nmType	Type of the Node Manager server. Valid values include: ■ `plain` for plain socket Java-based implementation ■ `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:/bea/user_projects/domains/oamdomain','plain')
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='plain')
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 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:/bea/mydomain/common/nodemanager/nm_password.properties. The nodemanager.domains file is written to *WL_HOME*/common/nodemanager by default.

```
wls:/mydomain/serverConfig> nmEnroll('c:/bea/mydomain/common/nodemanager')
Enrolling this machine with the domain directory at
c:\bea\mydomain\common\nodemanager....
Successfully enrolled this machine with the domain directory at
C:\bea\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:\bea\mydomain\servers\ms1\data\nodemanager\boot.properties
Successfully generated startup.properties at
c:\bea\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 `myServer`, 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])
```

Argument	Definition
`serverName`	Optional. Name of the server to be killed. This argument defaults to `myserver`.

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])
```

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.

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])
```

Argument	Definition
serverName	Optional. Name of the server for which you want to display the status. This argument defaults to `myserver`.

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 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.

3.10.10.2 Syntax

```
nmStart([serverName], [domainDir], [props], [writer])
```

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.

3.10.10.3 Example

The following example starts the `managed1` server in the current 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 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.

> **Note:** The WebLogic Server custom installation process optionally installs and starts Node Manager as a Windows service on Windows systems. For more information, see "Running the Installation Program in Graphical Mode" in the *Oracle WebLogic Server Installation Guide*. In this case, you do not need to start the Node Manager manually.

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:/bea/wlserver_10.3/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:\bea\wlserver_10.3\common\nodemanager.
wls:/mydomain/serverConfig>
```

3.11 Tree Commands

Use the WLST tree commands, listed in Table 3–13, 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

Table 3–12 (Cont.) Tree Commands for WLST Configuration

Use this command...	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

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 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 *Oracle Fusion Middleware 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 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 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 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 *Oracle Fusion Middleware 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 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 writeable hierarchy stores all of the configuration MBeans that represent your current 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 Fusion Middleware 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 Fusion Middleware 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
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. 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.	wls:/mydomain/edit> **cmo.setAdministrationPort(9092)**
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 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
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
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()**

Table 3–13 (Cont.) WLST Variables

Variable	Description	Example
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

4

Infrastructure Security Custom WLST Commands

The following sections describe the Oracle Fusion Middleware Infrastructure Security custom WLST commands in detail. Topics include:

- Section 4.1, "Overview of WSLT Security Commands"
- Section 4.2, "Audit Configuration Commands"
- Section 4.3, "SSL Configuration Commands"
- Section 4.4, "Oracle Identity Federation Commands"
- Section 4.5, "Directory Integration Platform Commands"
- Section 4.6, "Security Commands"

For additional information about Oracle Platform Security Services, see *Oracle Fusion Middleware Security Guide*.

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

4.1 Overview of WSLT Security Commands

WLST security commands are divided into the following categories:

Table 4–1 WLST Command Categories

Command Category	Description
Audit Configuration Commands	View and manage audit policies and the audit repository configuration
SSL Configuration Commands	View and manage wallets, JKS keystores, and SSL configuration for Oracle HTTP Server, Oracle WebCache, Oracle Internet Directory, and Oracle Virtual Directory components.
Oracle Identity Federation Commands	View and manage configuration for Oracle Identity Federation
Directory Integration Platform Commands	For information on DIP tools, see "Directory Integration Platform Tools" in the *Oracle Fusion Middleware User Reference for Oracle Identity Management*
Security Commands	Manage domain and credential domain stores and migrate domain policy store.

4.2 Audit Configuration Commands

Use the WLST commands listed in Table 4–2 to view and manage audit policies and the audit repository configuration.

Table 4–2 WLST Audit Commands

Use this command...	To...	Use with WLST...
getNonJavaEEAuditMBeanName	Display the mBean name for a non-Java EE component.	Online
getAuditPolicy	Display audit policy settings.	Online
setAuditPolicy	Update audit policy settings.	Online
getAuditRepository	Display audit repository settings.	Online
setAuditRepository	Update audit repository settings.	Online
listAuditEvents	List audit events for one or all components.	Online
exportAuditConfig	Export a component's audit configuration.	Online
importAuditConfig	Import a component's audit configuration.	Online

For more information, see the *Oracle Fusion Middleware Security Guide*.

4.2.1 getNonJavaEEAuditMBeanName

Online command that displays the mbean name for non-Java EE components.

4.2.1.1 Description

This command displays the mbean name for non-Java EE components given the instance name, component name, component type, and the name of the Oracle WebLogic Server on which the component's audit mbean is running. The mbean name is a required parameter to other audit WLST commands when managing a non-Java EE component.

4.2.1.2 Syntax

```
getNonJavaEEAuditMBeanName(instName, compName, compType, svrName)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are ohs, oid, ovd, and WebCache.
svrName	Specifies the name of the Oracle WebLogic Server.

4.2.1.3 Example

The following interactive command displays the mBean name for an Oracle Internet Directory:

```
wls:/mydomain/serverConfig> getNonJavaEEAuditMBeanName(instName='inst1',
compName='oid1', compType='oid', svrName='AdminServer')
```

4.2.2 getAuditPolicy

Online command that displays the audit policy settings.

4.2.2.1 Description

This command displays audit policy settings including the filter preset, special users, custom events, maximum log file size, and maximum log directory size. The component mbean name is required for non-Java EE components like Oracle Internet Directory and Oracle Virtual Directory.

> **Note:** You can obtain a non-Java EE component's MBean name using the getNonJavaEEAuditMBeanName command.

4.2.2.2 Syntax

```
getAuditPolicy([mbeanName])
```

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for non-Java EE components.

4.2.2.3 Examples

The following command displays the audit settings for a Java EE component:

```
wls:/mydomain/serverConfig> getAuditPolicy()
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean
as the root.
For more help, use help(domainRuntime)

FilterPreset:All
Max Log File Size:104857600
Max Log Dir Size:0
```

The following command displays the audit settings for MBean `CSAuditProxyMBean`:

```
wls:/mydomain/serverConfig>
getAuditPolicy(on='oracle.security.audit.test:type=CSAuditMBean,
name=CSAuditProxyMBean')
```

4.2.3 setAuditPolicy

Online command that updates an audit policy.

4.2.3.1 Description

Online command that configures the audit policy settings. You can set the filter preset, add or remove users, and add or remove custom events. The component mbean name is required for non-Java EE components like Oracle Internet Directory and Oracle Virtual Directory.

> **Note:** You can obtain a non-Java EE component's MBean name using the getNonJavaEEAuditMBeanName command.

4.2.3.2 Syntax

```
setAuditPolicy([mbeanName],[filterPreset],[addSpecialUsers],
[removeSpecialUsers],[addCustomEvents],[removeCustomEvents])
```

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for non-Java EE components.
filterPreset	Specifies the filter preset to be changed.
addSpecialUsers	Specifies the special users to be added.
removeSpecialUsers	Specifies the special users to be removed.
addCustomEvents	Specifies the custom events to be added.
removeCustomEvents	Specifies the custom events to be removed.

4.2.3.3 Examples

The following interactive command sets audit policy to `None` level, and adds users `user2` and `user3` while removing `user1` from the policy:

```
wls:/mydomain/serverConfig> setAuditPolicy
(filterPreset='None',addSpecialUsers='user2,user3',removeSpecialUsers='user1')

wls:/mydomain/serverConfig> getAuditPolicy();
Already in Domain Runtime Tree

FilterPreset:None
Special Users:user2,user3
Max Log File Size:104857600
Max Log Dir Size:0
```

The following interactive command adds login events while removing logout events from the policy:

```
wls:/mydomain/serverConfig>
setAuditPolicy(filterPreset='Custom',addCustomEvents='UserLogin',removeCustomEvent
s='UserLogout')
```

The following interactive command sets audit policy to a `Low` level:

```
wls:/IDMDomain/domainRuntime> setAuditPolicy(filterPreset='Low');
Already in Domain Runtime Tree
Audit Policy Information updated successfully

wls:/IDMDomain/domainRuntime> getAuditPolicy();
Already in Domain Runtime Tree
FilterPreset:Low
Max Log File Size:104857600
Max Log Dir Size:0
```

The following command sets a custom filter to audit the `CheckAuthorization` event:

```
wls:/IDMDomain/domainRuntime> setAuditPolicy(filterPreset='Custom',
addCustomEvents='JPS:CheckAuthorization');
Already in Domain Runtime Tree

Audit Policy Information updated successfully
```

```
wls:/IDMDomain/domainRuntime> getAuditPolicy();
Already in Domain Runtime Tree

FilterPreset:Custom
Special Users:user1
Max Log File Size:104857600
Max Log Dir Size:0
Custom Events:JPS:CheckAuthorization
```

4.2.4 getAuditRepository

Online command that displays audit repository settings.

4.2.4.1 Description

This command displays audit repository settings for Java EE components and applications (for other components like Oracle Internet Directory, the repository configuration resides in opmn.xml). Also displays database configuration if the repository is a database type.

4.2.4.2 Syntax

```
getAuditRepository
```

4.2.4.3 Example

The following command displays audit repository configuration:

```
wls:/IDMDomain/domainRuntime> getAuditRepository()
Already in Domain Runtime Tree

Repository Type:File
```

4.2.5 setAuditRepository

Online command that updates audit repository settings.

4.2.5.1 Description

This command sets the audit repository settings for Java EE components and applications (for other components like Oracle Internet Directory, the repository is configured by editing `opmn.xml`).

4.2.5.2 Syntax

```
setAuditRepository([switchToDB],[dataSourceName],[interval])
```

Argument	Definition
switchToDB	If `true`, switches the repository from file to database.
dataSourceName	Specifies the name of the data source.
interval	Specifies intervals at which the audit loader kicks off.

4.2.5.3 Examples

The following command switches from a file repository to a database repository:

```
wls:/IDMDomain/domainRuntime> setAuditRepository(switchToDB='true');
```

```
Already in Domain Runtime Tree

Audit Repository Information updated

wls:/IDMDomain/domainRuntime> getAuditRepository();
Already in Domain Runtime Tree

JNDI Name:jdbc/AuditDB
Interval:15
Repository Type:DB
```

The following interactive command changes audit repository to a specific database and sets the audit loader interval to 14 seconds:

```
wls:/mydomain/serverConfig>
setAuditRepository(switchToDB='true',dataSourceName='jdbcAuditDB',interval='14')
```

4.2.6 listAuditEvents

Online command that displays a component's audit events.

4.2.6.1 Description

This command displays a component's audit events and attributes. For non-Java EE components, pass the component mbean name as a parameter. Java EE applications and services like Oracle Platform Security Services (OPSS) do not need the mbean parameter. Without a component type, all generic attributes applicable to all components are displayed.

> **Note:** You can obtain a non-Java EE component's MBean name using the getNonJavaEEAuditMBeanName command.

4.2.6.2 Syntax

```
listAuditEvents([mbeanName],[componentType])
```

Argument	Definition
mbeanName	Specifies the name of the component MBean.
componentType	Specifies the component type.

4.2.6.3 Examples

The following command displays audit events for the Oracle Platform Security Services component:

```
wls:/IDMDomain/domainRuntime> listAuditEvents(componentType='JPS');
Already in Domain Runtime Tree

Common Attributes
ComponentType
Type of the component. For MAS integrated SystemComponents this is the
componentType
InstanceId
Name of the MAS Instance, that this component belongs to
HostId
DNS hostname of originating host
HostNwaddr
IP or other network address of originating host
```

```
ModuleId
ID of the module that originated the message. Interpretation is unique within
Component ID.
ProcessId
ID of the process that originated the message
```

The following command displays audit events for Oracle HTTP Server:

```
wls:/mydomain/serverConfig> listAuditEvents(componentType='ohs')
```

The following command displays all audit events:

```
wls:/IDMDomain/domainRuntime> listAuditEvents();
Already in Domain Runtime Tree

Components:
DIP
JPS
OIF
OWSM-AGENT
OWSM-PM-EJB
ReportsServer
WS-PolicyAttachment
WebCache
WebServices
Attributes applicable to all components:
ComponentType
InstanceId
HostId
HostNwaddr
ModuleId
ProcessId
OracleHome
HomeInstance
ECID
RID
...
```

4.2.7 exportAuditConfig

Online command that exports a component's audit configuration.

4.2.7.1 Description

This command exports the audit configuration to a file. For non-Java EE components, pass the component mbean name as a parameter. Java EE applications and services like Oracle Platform Security Services (OPSS) do not need the mbean parameter.

> **Note:** You can obtain a non-Java EE component's MBean name using the getNonJavaEEAuditMBeanName command.

4.2.7.2 Syntax

```
exportAuditConfig([mbeanName],fileName)
```

Argument	Definition
mbeanName	Specifies the name of the non-Java EE component MBean.

Argument	Definition
fileName	Specifies the path and file name to which the audit configuration should be exported.

4.2.7.3 Examples

The following interactive command exports the audit configuration for a component:

```
wls:/mydomain/serverConfig>
exportAuditConfig(on='oracle.security.audit.test:type=CSAuditMBean,name=CSAuditPro
xyMBean',fileName='/tmp/auditconfig')
```

The following interactive command exports the audit configuration for a Java EE component; no mBean is specified:

```
wls:/mydomain/serverConfig> exportAuditConfig(fileName='/tmp/auditconfig')
```

4.2.8 importAuditConfig

Online command that imports a component's audit configuration.

4.2.8.1 Description

This command imports the audit configuration from an external file. For non-Java EE components, pass the component mbean name as a parameter. Java EE applications and services like Oracle Platform Security Services (OPSS) do not need the mbean parameter.

> **Note:** You can obtain a non-Java EE component's MBean name using the getNonJavaEEAuditMBeanName command.

4.2.8.2 Syntax

```
importAuditConfig([mbeanName],fileName)
```

Argument	Definition
mbeanName	Specifies the name of the non-Java EE component MBean.
fileName	Specifies the path and file name from which the audit configuration should be imported.

4.2.8.3 Examples

The following interactive command imports the audit configuration for a component:

```
wls:/mydomain/serverConfig>
importAuditConfig(on='oracle.security.audit.test:type=CSAuditMBean,name='CSAuditPr
oxyMBean',fileName='/tmp/auditconfig')
```

The following interactive command imports the audit configuration for a component; no mBean is specified:

```
wls:/mydomain/serverConfig> importAuditConfig(fileName='/tmp/auditconfig')
```

4.3 SSL Configuration Commands

Use the WLST commands listed in Table 4–3 to view and manage SSL configuration for Oracle Fusion Middleware components.

Table 4–3 WLST Commands for SSL Configuration

Use this command...	To...	Use with WLST...
addCertificateRequest	Generate a certificate signing request in an Oracle wallet.	Online
addSelfSignedCertificate	Add a self-signed certificate to an Oracle wallet.	Online
changeKeyStorePassword	Change the password to a JKS keystore.	Online
changeWalletPassword	Change the password to an Oracle wallet.	Online
configureSSL	Set the SSL attributes for a component listener.	Online
createKeyStore	Create a JKS keystore.	Online
createWallet	Create an Oracle wallet.	Online
deleteKeyStore	Delete a JKS keystore.	Online
deleteWallet	Delete an Oracle wallet.	Online
exportKeyStore	Export a JKS keystore to a file.	Online
exportKeyStoreObject	Export an object from a JKS keystore to a file.	Online
exportWallet	Export an Oracle wallet to a file.	Online
exportWalletObject	Export an object from an Oracle wallet to a file.	Online
generateKey	Generate a keypair in a JKS keystore.	Online
getKeyStoreObject	Display a certificate or other object present in a JKS keystore.	Online
getSSL	Display the SSL attributes for a component listener.	Online
getWalletObject	Display a certificate or other object present in an Oracle wallet.	Online
importKeyStore	Import a JKS keystore from a file.	Online
importKeyStoreObject	Import a certificate or other object from a file to a JKS keystore.	Online
importWallet	Import an Oracle wallet from a file.	Online
importWalletObject	Import a certificate or other object from a file to an Oracle wallet.	Online
listKeyStoreObjects	List all objects present in a JKS keystore.	Online
listKeyStores	List all JKS keystores configured for a component instance.	Online
listWalletObjects	List all objects present in an Oracle wallet.	Online
listWallets	List all Oracle wallets configured for a component instance.	Online
removeKeyStoreObject	Remove a certificate or other object from a component instance's JKS keystore.	Online
removeWalletObject	Remove a certificate or other object from a component instance's Oracle wallet.	Online

For more information, see the *Oracle Fusion Middleware Administrator's Guide*.

4.3.1 addCertificateRequest

Online command that generates a certificate signing request in an Oracle wallet.

4.3.1.1 Description

This command generates a certificate signing request in Base64 encoded PKCS#10 format in an Oracle wallet for a component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). To get a certificate signed by a certificate authority (CA), send the certificate signing request to your CA.

4.3.1.2 Syntax

```
addCertificateRequest(instName, compName, compType, walletName, password, DN,
keySize)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
DN	Specifies the Distinguished Name of the key pair entry.
keySize	Specifies the key size in bits.

4.3.1.3 Example

The following command generates a certificate signing request with DN `cn=www.acme.com` and key size `1024` in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> addCertificateRequest('inst1', 'oid1',
'oid','wallet1', 'password', 'cn=www.acme.com', '1024',)
```

4.3.2 addSelfSignedCertificate

Online command that adds a self-signed certificate.

4.3.2.1 Description

This command creates a key pair and wraps it in a self-signed certificate in an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). Only keys based on the RSA algorithm are generated.

4.3.2.2 Syntax

```
addSelfSignedCertificate(instName, compName, compType, walletName, password, DN,
keySize)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.

Argument	Definition
DN	Specifies the Distinguished Name of the key pair entry.
keySize	Specifies the key size in bits.

4.3.2.3 Example

The following command adds a self-signed certificate with DN `cn=www.acme.com`, key size `1024` to `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance inst1:

```
wls:/mydomain/serverConfig> addSelfSignedCertificate('inst1', 'oid1',
'oid','wallet1', 'password', 'cn=www.acme.com', '1024')
```

4.3.3 changeKeyStorePassword

Online command that changes the keystore password.

4.3.3.1 Description

This command changes the password of a Java Keystore (JKS) file for an Oracle Virtual Directory instance.

4.3.3.2 Syntax

```
changeKeyStorePassword(instName, compName, compType, keystoreName, currPassword,
newPassword)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the filename of the keystore.
currPassword	Specifies the current keystore password.
newPassword	Specifies the new keystore password.

4.3.3.3 Example

The following command changes the password of file `keys.jks` for Oracle Virtual Directory instance `ovd1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> changeKeyStorePassword('inst1', 'ovd1',
'ovd','keys.jks', 'currpassword', 'newpassword')
```

4.3.4 changeWalletPassword

Online command that changes the password of an Oracle wallet.

4.3.4.1 Description

This command changes the password of an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). This command is only applicable to password-protected wallets.

4.3.4.2 Syntax

```
changeWalletPassword(instName, compName, compType, walletName,currPassword,
newPassword)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'oid', 'ohs', and 'webcache'.
walletName	Specifies the filename of the wallet.
currPassword	Specifies the current wallet password.
newPassword	Specifies the new wallet password.

4.3.4.3 Example

The following command changes the password for `wallet1` from `currpassword` to `newpassword` for Oracle HTTP Server instance `ohs1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> changeWalletPassword('inst1', 'ohs1', 'ohs','wallet1',
'currpassword', 'newpassword')
```

4.3.5 configureSSL

Online command that sets SSL attributes.

4.3.5.1 Description

This command sets the SSL attributes for a component listener. The attributes are specified in a properties file format (name=value). If a properties file is not provided, or it does not contain any SSL attributes, default attribute values are used. For component-specific SSL attribute value defaults, see the chapter "SSL Configuration in Oracle Fusion Middleware" in the *Oracle Fusion Middleware Administrator's Guide*.

4.3.5.2 Syntax

```
configureSSL(instName, compName, compType, listener, filePath)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'oid', 'ovd', ohs', and 'webcache'.
listener	Specifies the name of the component listener to be configured for SSL.
filePath	Specifies the absolute path of the properties file containing the SSL attributes to set.

4.3.5.3 Examples

The following command configures SSL attributes specified in the properties file `/tmp/ssl.properties` for Oracle Virtual Directory instance `ovd1` in application server instance `inst1`, for listener `listener1`:

```
wls:/mydomain/serverConfig> configureSSL('inst1', 'ovd1', 'ovd',
'listener1','/tmp/ssl.properties')
```

The following command configures SSL attributes without specifying a properties file. Since no file is provided, the default SSL attribute values are used:

```
wls:/mydomain/serverConfig> configureSSL('inst1', 'ovd1', 'ovd', 'listener2')
```

4.3.6 createKeyStore

Online command that creates a JKS keystore.

4.3.6.1 Description

This command creates a Java keystore (JKS) for the specified Oracle Virtual Directory instance. For keystore file location and other information, see the chapter "Managing Keystores, Wallets, and Certificates" in the *Oracle Fusion Middleware Administrator's Guide*.

4.3.6.2 Syntax

```
createKeyStore(instName, compName, compType, keystoreName, password)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the filename of the keystore file to be created.
password	Specifies the keystore password.

4.3.6.3 Example

The following command creates JKS file `keys.jks` with password `password` for Oracle Virtual Directory instance `ovd1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> createKeyStore('inst1', 'ovd1', 'ovd','keys.jks',
'password')
```

4.3.7 createWallet

Online command that creates an Oracle wallet.

4.3.7.1 Description

This command creates an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). Wallets can be of password-protected or auto-login type. For wallet details, see the chapter "Managing Keystores, Wallets, and Certificates" in the *Oracle Fusion Middleware Administrator's Guide*.

4.3.7.2 Syntax

```
createWallet(instName, compName, compType, walletName, password)
```

Argument	Definition
instName	Specifies the name of the application server instance.

Argument	Definition
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'oid', 'ohs', and 'webcache'.
walletName	Specifies the name of the wallet file to be created.
password	Specifies the wallet password.

4.3.7.3 Examples

The following command creates a wallet named `wallet1` with password `password`, for Oracle HTTP Server instance `ohs1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> createWallet('inst1', 'ohs1', 'ohs','wallet1',
'password')
```

The following command creates an auto-login wallet named `wallet2` for Oracle WebCache instance `wc1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> createWallet('inst1', 'wc1', 'webcache','wallet2', '')
```

4.3.8 deleteKeyStore

Online command that deletes a keystore.

4.3.8.1 Description

This command deletes a keystore for a specified Oracle Virtual Directory instance.

4.3.8.2 Syntax

```
deleteKeyStore(instName, compName, compType, keystoreName)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the name of the keystore file to delete.

4.3.8.3 Example

The following command deletes JKS file `keys.jks` for Oracle Virtual Directory instance `ovd1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> deleteKeyStore('inst1', 'ovd1', 'ovd','keys.jks')
```

4.3.9 deleteWallet

Online command that deletes an Oracle wallet.

4.3.9.1 Description

This command deletes an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory).

4.3.9.2 Syntax

```
deleteWallet(instName, compName, compType, walletName)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'oid', 'ohs', and 'webcache'.
walletName	Specifies the name of the wallet file to be deleted.

4.3.9.3 Example

The following command deletes a wallet named `wallet1` for Oracle HTTP Server instance `ohs1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> deleteWallet('inst1', 'ohs1', 'ohs','wallet1')
```

4.3.10 exportKeyStore

Online command that exports the keystore to a file.

4.3.10.1 Description

This command exports a keystore, configured for the specified Oracle Virtual Directory instance, to a file under the given directory. The exported filename is the same as the keystore name.

4.3.10.2 Syntax

```
exportKeyStore(instName, compName, compType, keystoreName, password, path)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the name of the keystore file.
password	Specifies the password of the keystore.
path	Specifies the absolute path of the directory under which the keystore is exported.

4.3.10.3 Example

The following command exports the keystore `keys.jks` for Oracle Virtual Directory instance `ovd1` to file `keys.jks` under `/tmp`:

```
wls:/mydomain/serverConfig> exportKeyStore('inst1', 'ovd1', 'ovd', 'keys.jks',
'password', '/tmp')
```

4.3.11 exportKeyStoreObject

Online command that exports an object from a keystore to a file.

4.3.11.1 Description

This command exports a certificate signing request, certificate/certificate chain, or trusted certificate present in a Java keystore (JKS) to a file for the specified Oracle Virtual Directory instance. The certificate signing request is generated before exporting the object. The alias specifies the object to be exported.

4.3.11.2 Syntax

```
exportKeyStoreObject(instName, compName, compType, keystoreName, password, type,
path, alias)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the name of the keystore file.
password	Specifies the password of the keystore.
type	Specifies the type of the keystore object to be exported. Valid values are 'CertificateRequest', 'Certificate', 'TrustedCertificate' and 'TrustedChain'.
path	Specifies the absolute path of the directory under which the object is exported as a file named base64.txt.
alias	Specifies the alias of the keystore object to be exported.

4.3.11.3 Examples

The following command generates and exports a certificate signing request from the key-pair indicated by alias `mykey` in `keys.jks`, for Oracle Virtual Directory instance `ovd1` in application server instance `inst1`. The certificate signing request is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportKeyStoreObject('inst1', 'ovd1',
'ovd','keys.jks', 'password', 'CertificateRequest', '/tmp','mykey')
```

The following command exports a certificate or certificate chain indicated by alias `mykey` in `keys.jks`, for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`. The certificate or certificate chain is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportKeyStoreObject('inst1', 'ovd1',
'ovd','keys.jks', 'password', 'Certificate', '/tmp','mykey')
```

The following command exports a trusted certificate indicated by alias `mykey` in `keys.jks`, for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`. The trusted certificate is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportKeyStoreObject('inst1', 'ovd1',
'ovd','keys.jks', 'password', 'TrustedCertificate', '/tmp','mykey')
```

4.3.12 exportWallet

Online command that exports an Oracle wallet.

4.3.12.1 Description

This command exports an Oracle wallet, configured for a specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory), to file(s) under the given directory. If the exported file is an auto-login only wallet, the file name is 'cwallet.sso'. If it is password-protected wallet, two files are created - 'ewallet.p12' and 'cwallet.sso'.

4.3.12.2 Syntax

```
exportWallet(instName, compName, compType, walletName,password, path)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'oid', 'ohs', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
path	Specifies the absolute path of the directory under which the object is exported.

4.3.12.3 Examples

The following command exports auto-login wallet `wallet1` for Oracle Internet Directory instance `oid1` to file `cwallet.sso` under `/tmp`:

```
wls:/mydomain/serverConfig> exportWallet('inst1', 'oid1', 'oid',
'wallet1','','/tmp')
```

The following command exports password-protected wallet `wallet2` for Oracle Internet Directory instance `oid1` to two files, `ewallet.p12` and `cwallet.sso`, under `/tmp`:

```
wls:/mydomain/serverConfig> exportWallet('inst1', 'oid1', 'oid', 'wallet2',
'password', '/tmp')
```

4.3.13 exportWalletObject

Online command that exports a certificate or other wallet object to a file.

4.3.13.1 Description

This command exports a certificate signing request, certificate, certificate chain or trusted certificate present in an Oracle wallet to a file for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). DN is used to indicate the object to be exported.

4.3.13.2 Syntax

```
exportWalletObject(instName, compName, compType, walletName, password, type, path,
DN)
```

Argument	Definition
instName	Specifies the name of the application server instance.

Argument	Definition
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
type	Specifies the type of wallet object to be exported. Valid values are 'CertificateRequest', 'Certificate', 'TrustedCertificate' or 'TrustedChain'.
path	Specifies the absolute path of the directory under which the object is exported as a file base64.txt.
DN	Specifies the Distinguished Name of the wallet object being exported.

4.3.13.3 Examples

The following command exports a certificate signing request with DN `cn=www.acme.com` in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`. The certificate signing request is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'CertificateRequest', '/tmp','cn=www.acme.com')
```

The following command exports a certificate with DN `cn=www.acme.com` in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`. The certificate or certificate chain is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'Certificate', '/tmp','cn=www.acme.com')
```

The following command exports a trusted certificate with DN `cn=www.acme.com` in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`. The trusted certificate is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'TrustedCertificate', '/tmp','cn=www.acme.com')
```

The following command exports a certificate chain with DN `cn=www.acme.com` in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`. The certificate or certificate chain is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'TrustedChain', '/tmp','cn=www.acme.com')
```

4.3.14 generateKey

Online command that generates a key pair in a Java keystore.

4.3.14.1 Description

This command generates a key pair in a Java keystore (JKS) for Oracle Virtual Directory. It also wraps the key pair in a self-signed certificate. Only keys based on the RSA algorithm are generated.

4.3.14.2 Syntax

```
generateKey(instName, compName, compType, keystoreName, password, DN, keySize,
```

```
alias, algorithm)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the name of the keystore.
password	Specifies the password of the keystore.
DN	Specifies the Distinguished Name of the key pair entry.
keySize	Specifies the key size in bits.
alias	Specifies the alias of the key pair entry in the keystore.
algorithm	Specifies the key algorithm. Valid value is 'RSA'.

4.3.14.3 Examples

The following command generates a key pair with DN `cn=www.acme.com`, key size `1024`, algorithm `RSA` and alias `mykey` in `keys.jks`, for Oracle Virtual Directory instance `ovd1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> generateKey('inst1', 'ovd1', 'ovd','keys.jks',
'password', 'cn=www.acme.com', '1024', 'mykey', 'RSA')
```

The following command is the same as above, except it does not explicitly specify the key algorithm:

```
wls:/mydomain/serverConfig> generateKey('inst1', 'ovd1', 'ovd','keys.jks',
'password', 'cn=www.acme.com', '1024', 'mykey')
```

4.3.15 getKeyStoreObject

Online command that shows details about a keystore object.

4.3.15.1 Description

This command displays a specific certificate or trusted certificate present in a Java keystore (JKS) for Oracle Virtual Directory. The keystore object is indicated by its index number, as given by the `listKeyStoreObjects` command. It shows the certificate details including DN, key size, algorithm, and other information.

4.3.15.2 Syntax

```
getKeyStoreObject(instName, compName, compType, keystoreName, password, type,
index)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the name of the keystore file.
password	Specifies the password of the keystore.

Argument	Definition
type	Specifies the type of the keystore object to be listed. Valid values are 'Certificate' and 'TrustedCertificate'.
index	Specifies the index number of the keystore object as returned by the `listKeyStoreObjects` command.

4.3.15.3 Examples

The following command shows a trusted certificate with index `1` present in `keys.jks`, for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> getKeyStoreObject('inst1', 'ovd1', 'ovd','keys.jks',
'password', 'TrustedCertificate', '1')
```

The following command shows a certificate with index `1` present in `keys.jks`, for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> getKeyStoreObject('inst1', 'ovd1', 'ovd','keys.jks',
'password', 'Certificate', '1')
```

4.3.16 getSSL

Online command that lists the configured SSL attributes.

4.3.16.1 Description

This command lists the configured SSL attributes for the specified component listener. For Oracle Internet Directory, the listener name is always `sslport1`.

4.3.16.2 Syntax

```
getSSL(instName, compName, compType, listener)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'ovd', 'oid', 'ohs', and 'webcache'.
listener	Specifies the name of the component listener.

4.3.16.3 Example

The following command shows the SSL attributes configured for Oracle Internet Directory instance `oid1`, in application server instance `inst1`, for listener `sslport1`:

```
wls:/mydomain/serverConfig> getSSL('inst1', 'oid1', 'oid', 'sslport1')
```

4.3.17 getWalletObject

Online command that displays information about a certificate or other object in an Oracle wallet.

4.3.17.1 Description

This command displays a specific certificate signing request, certificate or trusted certificate present in an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). The wallet object is indicated by its index number, as given by the `listWalletObjects` command. For certificates or trusted certificates, it shows the certificate details including DN, key size, algorithm and other data. For certificate signing requests, it shows the subject DN, key size and algorithm.

4.3.17.2 Syntax

```
getWalletObject(instName, compName, compType, walletName, password, type, index)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
type	Specifies the type of wallet object to be exported. Valid values are 'CertificateRequest', 'Certificate', and 'TrustedCertificate'.
index	Specifies the index number of the wallet object as returned by the `listWalletObjects` command.

4.3.17.3 Examples

The following command shows certificate signing request details for the object with index `0` present in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> getKeyStoreObject('inst1', 'oid1',
'oid','wallet1','password', 'CertificateRequest', '0')
```

The following command shows certificate details for the object with index `0` present in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> getKeyStoreObject('inst1', 'oid1',
'oid','wallet1','password', 'Certificate', '0')
```

The following command shows trusted certificate details for the object with index `0`, present in `wallet1`, for Oracle Internet Directory instance `oid1`, in application serverinstance `inst1`:

```
wls:/mydomain/serverConfig> getKeyStoreObject('inst1', 'oid1',
'oid','wallet1','password', 'TrustedCertificate', '0')
```

4.3.18 importKeyStore

Online command that imports a keystore from a file.

4.3.18.1 Description

This command imports a Java keystore (JKS) from a file to the specified Oracle Virtual Directory instance for manageability. The component instance name must be unique.

4.3.18.2 Syntax

```
importKeyStore(instName, compName, compType, keystoreName, password, filePath)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the name of the keystore being imported. This name must be unique for this component instance.
password	Specifies the password of the keystore.
filePath	Specifies the absolute path of the keystore file to be imported.

4.3.18.3 Example

The following command imports the keystore `/tmp/keys.jks` as `file.jks` into Oracle Virtual Directory instance `ovd1`. Subsequently, the keystore is managed through the name `file.jks`:

```
wls:/mydomain/serverConfig> importKeyStore('inst1', 'ovd1', 'ovd', 'file.jks',
'password', '/tmp/keys.jks')
```

4.3.19 importKeyStoreObject

Online command that imports an object from a file to a keystore.

4.3.19.1 Description

This command imports a certificate, certificate chain, or trusted certificate into a Java keystore (JKS) for Oracle Virtual Directory, assigning it the specified alias which must be unique in the keystore. If a certificate or certificate chain is being imported, the alias must match that of the corresponding key-pair.

4.3.19.2 Syntax

```
importKeyStoreObject(instName, compName, compType, keystoreName, password, type,
filePath, alias)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the name of the keystore.
password	Specifies the password of the keystore.
type	Specifies the type of the keystore object to be imported. Valid values are 'Certificate' and 'TrustedCertificate'.
filePath	Specifies the absolute path of the file containing the keystore object.
alias	Specifies the alias to assign to the keystore object to be imported.

4.3.19.3 Examples

The following command imports a certificate or certificate chain from file `cert.txt` into `keys.jks`, using alias `mykey` for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`. The file `keys.jks` must already have an alias `mykey` for a key-pair whose public key matches that in the certificate being imported:

```
wls:/mydomain/serverConfig> > importKeyStoreObject('inst1', 'ovd1',
'ovd','keys.jks', 'password', 'Certificate','/tmp/cert.txt', 'mykey')
```

The following command imports a trusted certificate from file `trust.txt` into `keys.jks` using alias `mykey1`, for Oracle Virtual Directory instance `ovd1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> importKeyStoreObject('inst1', 'ovd1',
'ovd','keys.jks', 'password', 'TrustedCertificate','/tmp/trust.txt', 'mykey1')
```

4.3.20 importWallet

Online command that imports an Oracle wallet from a file.

4.3.20.1 Description

This command imports an Oracle wallet from a file to the specified component instance (Oracle HTTP Server, Oracle WebCache, or Oracle Internet Directory) for manageability. If the wallet being imported is an auto-login wallet, the file path must point to `cwallet.sso`; if the wallet is password-protected, it must point to `ewallet.p12`. The wallet name must be unique for the component instance.

4.3.20.2 Syntax

```
importWallet(instName, compName, compType, walletName, password, filePath)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet being imported. The name must be unique for the component instance.
password	Specifies the password of the wallet.
filePath	Specifies the absolute path of the wallet file being imported.

4.3.20.3 Examples

The following command imports auto-login wallet file `/tmp/cwallet.sso` as `wallet1` into Oracle Internet Directory instance `oid1`. Subsequently, the wallet is managed with the name `wallet1`. No password is passed since it is an auto-login wallet:

```
wls:/mydomain/serverConfig> importWallet('inst1', 'oid1', 'oid', 'wallet1', '',
'/tmp/cwallet.sso')
```

The following command imports password-protected wallet `/tmp/ewallet.p12` as `wallet2` into Oracle Internet Directory instance `oid1`. Subsequently, the wallet is managed with the name `wallet2`. The wallet password is passed as a parameter:

```
wls:/mydomain/serverConfig> importWallet('inst1', 'oid1', 'oid', 'wallet2',
'password', '/tmp/ewallet.p12')
```

4.3.21 importWalletObject

Online command that imports a certificate or other object into an Oracle wallet.

4.3.21.1 Description

This command imports a certificate, trusted certificate or certificate chain into an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache component or Oracle Internet Directory).When importing a certificate, use the same wallet file from which the certificate signing request was generated.

4.3.21.2 Syntax

```
importWalletObject(instName, compName, compType, walletName, password, type,
filePath)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
type	Specifies the type of wallet object to be imported. Valid values are 'Certificate', 'TrustedCertificate' and 'TrustedChain'.
filePath	Specifies the absolute path of the file containing the wallet object.

4.3.21.3 Examples

The following command imports a certificate chain in PKCS#7 format from file `chain.txt` into `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> importWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'TrustedChain','/tmp/chain.txt')
```

The following command imports a certificate from file `cert.txt` into `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> importWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'Certificate','/tmp/cert.txt')
```

The following command imports a trusted certificate from file `trust.txt` into `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> importWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'TrustedCertificate','/tmp/trust.txt')
```

4.3.22 listKeyStoreObjects

Online command that lists the contents of a keystore.

4.3.22.1 Description

This command lists all the certificates or trusted certificates present in a Java keystore (JKS) for Oracle Virtual Directory.

4.3.22.2 Syntax

```
listKeyStoreObjects(instName, compName, compType, keystoreName, password, type)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the name of the keystore file.
password	Specifies the password of the keystore.
type	Specifies the type of keystore object to be listed. Valid values are 'Certificate' and 'TrustedCertificate'.

4.3.22.3 Examples

The following command lists all trusted certificates present in `keys.jks`, for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> listKeyStoreObjects('inst1', 'ovd1', 'ovd','keys.jks',
'password', 'TrustedCertificate')
```

The following command lists all certificates present in `keys.jks`, for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> listKeyStoreObjects('inst1', 'ovd1', 'ovd','keys.jks',
'password', 'Certificate')
```

4.3.23 listKeyStores

Online command that lists all the keystores for a component.

4.3.23.1 Description

This command lists all the Java keystores (JKS) configured for the specified Oracle Virtual Directory instance.

4.3.23.2 Syntax

```
listKeyStores(instName, compName, compType)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance
compType	Specifies the type of component. Valid value is 'ovd'.

4.3.23.3 Example

The following command lists all keystores for Oracle Virtual Directory instance `ovd1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> listKeyStores('inst1', 'ovd1', 'ovd')
```

4.3.24 listWalletObjects

Online command that lists all objects in an Oracle wallet.

4.3.24.1 Description

This command lists all certificate signing requests, certificates, or trusted certificates present in an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory).

4.3.24.2 Syntax

```
listWalletObjects(instName, compName, compType, walletName, password, type)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
type	Specifies the type of wallet object to be listed. Valid values are 'CertificateRequest', 'Certificate', and 'TrustedCertificate'.

4.3.24.3 Examples

The following command lists all certificate signing requests in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> > listWalletObjects('inst1', 'oid1',
'oid','wallet1','password', 'CertificateRequest')
```

The following command lists all certificates in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> listWalletObjects('inst1', 'oid1',
'oid','wallet1','password', 'Certificate')
```

The following command lists all trusted certificates in `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> listWalletObjects('inst1', 'oid1',
'oid','wallet1','password', 'TrustedCertificate')
```

4.3.25 listWallets

Online command that lists all wallets configured for a component instance.

4.3.25.1 Description

This command displays all the wallets configured for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory), and identifies the auto-login wallets.

4.3.25.2 Syntax

```
listWallets(instName, compName, compType)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.

4.3.25.3 Example

The following command lists all wallets for Oracle Internet Directory instance `oid1` in application server instance `inst1`:

```
wls:/mydomain/serverConfig> listWallets('inst1', 'oid1', 'oid')
```

4.3.26 removeKeyStoreObject

Online command that removes an object from a keystore.

4.3.26.1 Description

This command removes a certificate request, certificate, trusted certificate, or all trusted certificates from a Java keystore (JKS) for Oracle Virtual Directory. Use an alias to remove a specific object; no alias is needed if all trusted certificates are being removed.

4.3.26.2 Syntax

```
removeKeyStoreObject(instName, compName, compType, keystoreName, password, type,
alias)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ovd'.
keystoreName	Specifies the name of the keystore file.
password	Specifies the password of the keystore.
type	Specifies the type of the keystore object to be removed. Valid values are 'Certificate', 'TrustedCertificate' or 'TrustedAll'.
alias	Specifies the alias of the keystore object to be removed.

4.3.26.3 Examples

The following command removes a certificate or certificate chain denoted by alias `mykey` in `keys.jks`, for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> removeKeyStoreObject('inst1', 'ovd1',
'ovd','keys.jks', 'password', 'Certificate','mykey')
```

The following command removes a trusted certificate denoted by alias `mykey` in `keys.jks`, for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> removeKeyStoreObject('inst1', 'ovd1',
```

```
'ovd','keys.jks', 'password', 'TrustedCertificate','mykey')
```

The following command removes all trusted certificates in `keys.jks`, for Oracle Virtual Directory instance `ovd1`, in application server instance `inst1`. Since no alias is required, the value `None` is passed for that parameter:

```
wls:/mydomain/serverConfig> removeKeyStoreObject('inst1', 'ovd1',
'ovd','keys.jks', 'password', 'TrustedAll',None)
```

4.3.27 removeWalletObject

Online command that removes a certificate or other object from an Oracle wallet.

4.3.27.1 Description

This command removes a certificate signing request, certificate, trusted certificate or all trusted certificates from an Oracle wallet for the specified component instance (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). DN is used to indicate the object to be removed.

4.3.27.2 Syntax

```
removeWalletObject(instName, compName, compType, walletName, password, type, DN)
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
type	Specifies the type of the keystore object to be removed. Valid values are 'CertificateRequest', 'Certificate', 'TrustedCertificate' or 'TrustedAll'.
DN	Specifies the Distinguished Name of the wallet object to be removed.

4.3.27.3 Examples

The following command removes all trusted certificates from `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`. It is not necessary to provide a DN, so we pass null (denoted by `None`) for the DN parameter:

```
wls:/mydomain/serverConfig> removeWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'TrustedAll',None)
```

The following command removes a certificate signing request indicated by DN `cn=www.acme.com` from `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> removeWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'CertificateRequest','cn=www.acme.com')
```

The following command removes a certificate indicated by DN `cn=www.acme.com` from `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> removeWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'Certificate','cn=www.acme.com')
```

The following command removes a trusted certificate indicated by DN `cn=www.acme.com` from `wallet1`, for Oracle Internet Directory instance `oid1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> removeWalletObject('inst1', 'oid1', 'oid','wallet1',
'password', 'TrustedCertificate','cn=www.acme.com')
```

4.4 Oracle Identity Federation Commands

Use the WLST commands listed in Table 4–4 to view and manage configuration for Oracle Identity Federation.

Table 4–4 WLST Commands for Oracle Identity Federation

Use this command...	To...	Use with WLST...
addConfigListEntryInMap	Add a configuration list entry to a map.	Online
addConfigMapEntryInMap	Add a configuration map entry to a map.	Online
addConfigPropertyListEntry	Add a configuration property list entry.	Online
addConfigPropertyMapEntry	Add a configuration property map entry to the map.	Online
addCustomAuthnEngine	Add a custom authentication engine.	Online
addCustomSPEngine	Add a custom SP engine.	Online
addFederationListEntryInMap	Add a federations list entry to the map.	Online
addFederationMapEntryInMap	Add a federation map entry to the map.	Online
addFederationPropertyListEntry	Add a federation property list entry.	Online
addFederationPropertyMapEntry	Add a federation property map entry.	Online
deleteCustomAuthnEngine	Delete a custom authentication engine.	Online
deleteCustomSPEngine	Delete a custom SP engine.	Online
deleteProviderFederation	Delete a provider from the federation.	Online
deleteUserFederations	Delete a user from the federation.	Online
changeMessageStore	Change the message store to memory or RDBMS.	Online
changePeerProviderDescription	Change a peer provider's description.	Online
changeSessionStore	Change the session store to memory or RDBMS.	Online
createConfigPropertyList	Create a configuration property list.	Online
createConfigPropertyListInMap	Create a configuration property list in the map.	Online
createConfigPropertyMap	Create a configuration property map.	Online
createConfigPropertyMapInMap	Create a nested configuration property map in a map.	Online
createFederationPropertyList	Create a federation property list.	Online

Table 4–4 (Cont.) WLST Commands for Oracle Identity Federation

Use this command...	To...	Use with WLST...
createFederationPropertyListInMap	Create a federation property list in the map.	Online
createFederationPropertyMap	Create a federation property map.	Online
createFederationPropertyMapInMap	Create a nested federation property map in a map.	Online
createPeerProviderEntry	Create a peer provider entry.	Online
getConfigListValueInMap	Retrieve a configuration list value from the map.	Online
getConfigMapEntryInMap	Retrieve a configuration map value from the map.	Online
getConfigProperty	Retrieve a configuration property entry.	Online
getConfigPropertyList	Retrieve a configuration property list.	Online
getConfigPropertyMapEntry	Retrieve a configuration property map entry.	Online
getFederationListValueInMap	Retrieve a federation list value from the map.	Online
getFederationMapEntryInMap	Retrieve a federation map entry from a nested map.	Online
getFederationProperty	Retrieve a federation property.	Online
getFederationPropertyList	Retrieve the federation property list.	Online
getFederationPropertyMapEntry	Retrieve a federation property map entry.	Online
listCustomAuthnEngines	Display the list of custom authentication engines.	Online
listCustomSPEngines	Display the list of custom SP engines.	Online
loadMetadata	Load metadata from a file.	Online
removeConfigListInMap	Delete a configuration list in the map.	Online
removeConfigMapEntryInMap	Delete a configuration map entry in the map.	Online
removeConfigMapInMap	Delete a nested configuration map.	Online
removeConfigProperty	Delete a configuration property.	Online
removeConfigPropertyList	Delete a property list.	Online
removeConfigPropertyMap	Delete a property map.	Online
removeConfigPropertyMapEntry	Delete an entry in the property map.	Online
removeFederationListInMap	Delete a federation list in the map.	Online
removeFederationMapInMap	Delete a nested federation map.	Online
removeFederationMapEntryInMap	Delete a nested federation map entry.	Online
removeFederationProperty	Delete a federation property.	Online
removeFederationPropertyList	Delete a federation property list.	Online
removeFederationPropertyMap	Delete a federation property map.	Online
removeFederationPropertyMapEntry	Delete a federation property map entry.	Online

Table 4–4 (Cont.) WLST Commands for Oracle Identity Federation

Use this command...	To...	Use with WLST...
removePeerProviderEntry	Delete a peer provider entry.	Online
setConfigProperty	Set a configuration property.	Online
setCustomAuthnEngine	Define a custom authentication engine.	Online
setCustomSPEngine	Define a custom SP engine.	Online
setFederationProperty	Set a federation property.	Online

For more information, see the *Oracle Fusion Middleware Administrator's Guide for Oracle Identity Federation*.

4.4.1 addConfigListEntryInMap

Online command that adds a property value to a map.

4.4.1.1 Description

This command adds a property value to a nested list inside a map in config.xml.

4.4.1.2 Syntax

```
addConfigListEntryInMap(configName, mapname, listName, value, type)
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
mapname	Specifies the name of the property to map to be changed in config.xml.
listname	Specifies the name of the list.
value	Specifies the property value.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.1.3 Example

The following command adds `valueA` to a map list in server configuration:

```
wls:/mydomain/serverConfig>
addConfigListEntryInMap('serverconfig','mymap','mylistA','valueA','string')
```

4.4.2 addConfigMapEntryInMap

Online command that adds a nested map property entry in a map.

4.4.2.1 Description

This command that adds a property name/value pair to a map nested inside a map in config.xml.

4.4.2.2 Syntax

```
addConfigMapEntryInMap(configName, mapname, nestedMapName, propName, value, type)
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
mapname	Specifies the name of the property map to be changed in config.xml.
nestedmapname	name of the nested property map to be changed.
propname	Specifies the name of the list.
value	Specifies the property value.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.2.3 Example

The following command adds a boolean name/value pair to `nestedmapB` inside the map `mymap`.

```
wls:/mydomain/serverConfig>
addConfigMapEntryInMap('serverconfig','mymap','nestedmapB','myvarB','true',
'boolean')
```

4.4.3 addConfigPropertyListEntry

Online command that adds a list property entry to config.xml.

4.4.3.1 Description

This command adds a property value to a list in config.xml.

4.4.3.2 Syntax

```
addConfigPropertyListEntry(configName, listName, value, type)
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
listname	Specifies the name of the property list to be added in config.xml.
value	Specifies the new property list value. The entered value is appended to the list.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.3.3 Example

The following command adds a string value to `mylistA`.

```
wls:/mydomain/serverConfig>
addConfigPropertyListEntry('serverconfig','mylistA','valueA','string')
```

4.4.4 addConfigPropertyMapEntry

Online command that adds a property name/value entry in a map in config.xml.

4.4.4.1 Description

This command adds a property name/value entry in a map in config.xml.

4.4.4.2 Syntax

```
addConfigPropertyMapEntry(configName, mapName, propName, value, type)
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
mapname	Specifies the name of the property map in config.xml.
propname	Specifies the name of the property map.
value	Specifies the property map value to be added.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.4.3 Example

The following command adds `valueA` of string type to a map.

```
wls:/mydomain/serverConfig>
addConfigPropertyMapEntry('serverconfig','mymapA','myvarA','valueA','string')
```

4.4.5 addCustomAuthnEngine

Online command that adds a custom authentication integration engine.

4.4.5.1 Description

This command adds a custom authentication integration engine to config.xml.

4.4.5.2 Syntax

```
addCustomAuthnEngine(name, [enabled], [webContext], [authnRelativePath],
[logoutRelativePath], [logoutEnabled])
```

Argument	Definition
name	Specifies the name of the custom engine.
enabled	This flag specifies whether the engine is enabled (true) or not (false, default).
webContext	Specifies the web context for the engine.
authnRelativePath	Specifies the authentication relative path URL for the engine.
logoutRelativePath	Specifies the logout relative path URL for the engine.
logoutEnabled	This flag is set true to enable logout for the engine, else false.

4.4.5.3 Example

The following command defines an engine named `test` and enables it.

```
wls:/mydomain/serverConfig> addCustomAuthnEngine('test','true')
```

4.4.6 addCustomSPEngine

Online command that adds a custom service provider (SP) engine.

4.4.6.1 Description

This command adds a custom SP integration engine to config.xml.

4.4.6.2 Syntax

```
addCustomSPEngine(name, [enabled, [authnMech], [webContext], [authnRelativePath],
[logoutRelativePath], [logoutEnabled])
```

Argument	Definition
name	Specifies the name of the custom engine.
enabled	This flag specifies whether the engine is enabled (true) or not (false).
authnMech	Specifies the authentication mechanism for the engine.
webContext	Specifies the web context for the engine.
authnRelativePath	Specifies the authentication relative path URL for the engine.
logoutRelativePath	Specifies the logout relative path URL for the engine.
logoutEnabled	This flag is set true to enable logout for the engine, else false.

4.4.6.3 Example

The following command adds an engine and gives it a disabled status.

```
addCustomSPEngine('new
engine','false','oracle:fed:authentication:unspecified','webcontext')
```

4.4.7 addFederationListEntryInMap

Online command that adds a list property entry in a map.

4.4.7.1 Description

This command adds a property value to a nested list inside a map in cot.xml.

4.4.7.2 Syntax

```
addFederationListEntryInMap(providerID, mapname, listName, value, type)
```

Argument	Definition
providerID	Specifies the provider ID.
mapname	Specifies the name of the property map to be changed in cot.xml.
listname	Specifies the name of the property list to be added to the map.
value	Specifies the property list value to be added. The entered value is appended to the list.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.7.3 Example

The following command adds a boolean property list to `mymap`.

```
wls:/mydomain/serverConfig>
addFederationListEntryInMap('providerB','mymap','mylistB','true','boolean')
```

4.4.8 addFederationMapEntryInMap

Online command that adds a nested map property entry in a map.

4.4.8.1 Description

This command adds a property name/value pair to a map nested inside a map in cot.xml.

4.4.8.2 Syntax

```
addFederationMapEntryInMap(providerID, mapname, nestedMapName, propName, value,
type)
```

Argument	Definition
providerID	Specifies the provider ID.
mapname	Specifies the name of the property map to be changed in cot.xml.
nestedMapName	Specifies the name of the nested property map to be changed.
propName	Specifies the name of the property to be updated in the map.
value	Specifies the property value to be added. The entered value is appended to the list.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.8.3 Example

The following command adds a value of type string to the `myvarA` property in a nested map.

```
wls:/mydomain/serverConfig>
addFederationMapEntryInMap('providerA','mymap','nestedmapA','myvarA','valueA',
'string')
```

4.4.9 addFederationPropertyListEntry

Online command that adds a list property entry.

4.4.9.1 Description

This command adds a property value to a list in cot.xml.

4.4.9.2 Syntax

```
addFederationPropertyListEntry(providerID, listName, value, type)
```

Argument	Definition
providerID	Specifies the provider ID.
listName	Specifies the name of the property list to be updated.
value	Specifies the property list value to be added. The entered value is appended to the list.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.9.3 Example

The following command adds a value in string format to a specified property list.

```
wls:/mydomain/serverConfig>
addFederationPropertyListEntry('providerA','mylistA','valueA','string')
```

4.4.10 addFederationPropertyMapEntry

Online command that a property name/value entry in a map.

4.4.10.1 Description

This command adds a property name/value pair to a map in cot.xml.

4.4.10.2 Syntax

```
addFederationPropertyMapEntry(providerID, mapName, propName, value, type)
```

Argument	Definition
providerID	Specifies the provider ID.
mapname	Specifies the name of the property map to be changed in cot.xml.
propName	Specifies the name of the property to be added in the map.
value	Specifies the property value to be added. The entered value is appended to the list.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.10.3 Example

The following command adds boolean property `myvarB` to a map.

```
wls:/mydomain/serverConfig>
addFederationPropertyMapEntry('providerA','mymapB','myvarB','true','boolean')
```

4.4.11 deleteCustomAuthnEngine

Online command that deletes a custom authentication integration engine from the configuration.

4.4.11.1 Description

This command deletes a custom authentication integration engine in config.xml. You must provide the engine ID for an existing custom authentication engine in config.xml.

4.4.11.2 Syntax

```
deleteCustomAuthnEngine(engineID)
```

Argument	Definition
engineID	Specifies the engine ID of an existing engine to be deleted.

4.4.11.3 Example

The following command deletes the authentication engine with ID `id1234`.

```
wls:/mydomain/serverConfig> deleteCustomAuthnEngine('id1234')
```

4.4.12 deleteCustomSPEngine

Online command that deletes a custom service provider (SP) integration engine from the configuration.

4.4.12.1 Description

This command deletes a custom SP integration engine in config.xml. The EngineID for an existing custom SP engine in config.xml must be provided.

4.4.12.2 Syntax

```
ddeleteCustomSPEngine(engineID)
```

Argument	Definition
engineID	Specifies the engine ID of an existing engine to be deleted.

4.4.12.3 Example

The following command deletes the engine with ID `id1234`.

```
wls:/mydomain/serverConfig> deleteCustomSPEngine('id1234')
```

4.4.13 deleteProviderFederation

Online command that deletes federations for given provider.

4.4.13.1 Description

This command deletes federations for given provider ID.

4.4.13.2 Syntax

```
deleteProviderFederation(providerID)
```

Argument	Definition
providerID	Specifies the ProviderID for the peer provider for which federation is to be deleted.

4.4.13.3 Example

The following command deletes `providerA`:

```
wls:/mydomain/serverConfig> deleteProviderFederation(providerA)
```

4.4.14 deleteUserFederations

Online command that deletes federations for given users.

4.4.14.1 Description

This command deletes federations for the given list of users.

4.4.14.2 Syntax

```
deleteUserFederations([user1,..])
```

Argument	Definition
user1	Specifies a comma-separated list of users whose federations are to be deleted. At least one user must be specified.

4.4.14.3 Example

The following command deletes federations for three users:

```
wls:/mydomain/serverConfig> deleteUserFederations(['userA','userB','userC'])
```

4.4.15 changeMessageStore

Online command that changes the message store between memory and RDBMS.

4.4.15.1 Description

This command changes the message store to memory or RDBMS.

4.4.15.2 Syntax

```
changeMessageStore(type, [jndiname])
```

Argument	Definition
type	Specifies the type of store, RDBMS or Memory. Default is Memory.
jndiname	Specifies the `jndi` name to set for the store. Required if type is RDBMS.

4.4.15.3 Example

The following command changes the message store to RDBMS:

```
wls:/mydomain/serverConfig> changeMessageStore('RDBMS','jdbc/mydb')
```

4.4.16 changePeerProviderDescription

Online command that changes the peer provider description.

4.4.16.1 Description

This command updates a peer provider description in cot.xml.

4.4.16.2 Syntax

```
changePeerProviderDescription(providerID, description)
```

Argument	Definition
providerID	Specifies the provider ID.
description	Specifies the provider description.

4.4.16.3 Example

The following command updates the description of a provider:

```
wls:/mydomain/serverConfig> changePeerProviderDescription('providerA','new
description')
```

4.4.17 changeSessionStore

Online command that changes the session store between memory and RDBMS.

4.4.17.1 Description

This command changes the session store to memory or RDBMS.

4.4.17.2 Syntax

```
changeSessionStore(type, [jndiname])
```

Argument	Definition
type	Specifies the type of store, RDBMS or Memory. Default is Memory.
jndiname	Specifies the jndi name to set for the store. Required if type is RDBMS.

4.4.17.3 Example

The following command changes the session store to RDBMS.

```
wls:/mydomain/serverConfig> changeSessionStore('RDBMS','jdbc/mydb')
```

4.4.18 createConfigPropertyList

Online command that creates a property list.

4.4.18.1 Description

This command creates a property list in config.xml.

4.4.18.2 Syntax

```
createConfigPropertyList(configName, listName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
listName	Specifies the property list name.

4.4.18.3 Example

The following command creates property list `mylistA`.

```
wls:/mydomain/serverConfig> createConfigPropertyList('serverconfig','mylistA')
```

4.4.19 createConfigPropertyListInMap

Online command that creates a property list nested in the property map.

4.4.19.1 Description

This command creates a property list, nested in the property map, in config.xml.

4.4.19.2 Syntax

```
createConfigPropertyListInMap(configName, mapName, listName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20,..) to be updated.
mapName	Specifies an existing property map to contain the nested list.
listName	Specifies the property list name.

4.4.19.3 Example

The following command creates property list `mylistA` nested in a property map.

```
wls:/mydomain/serverConfig>
createConfigPropertyListInMap('serverconfig','mymapA','mylistA')
```

4.4.20 createConfigPropertyMap

Online command that creates a property map.

4.4.20.1 Description

This command that creates a property map in config.xml.

4.4.20.2 Syntax

```
createConfigPropertyMap(configName, mapName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
mapName	Specifies the property map to create.

4.4.20.3 Example

The following command creates property map `mymapA`:

```
wls:/mydomain/serverConfig> createConfigPropertyMap('serverconfig','mymapA')
```

4.4.21 createConfigPropertyMapInMap

Online command that creates a property map.

4.4.21.1 Description

This command that creates a property map in config.xml.

4.4.21.2 Syntax

```
createConfigPropertyMapInMap(configName, mapName, nestedMapName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
mapName	Specifies the name of an existing property map.
nestedMapName	Specifies the name of the property map to create nested inside mapName.

4.4.21.3 Example

The following command creates nested property map `nestedmymapA`:

```
wls:/mydomain/serverConfig>
createConfigPropertyMapInMap('serverconfig','mymapA','nestedmapA')
```

4.4.22 createFederationPropertyList

Online command that creates a property list.

4.4.22.1 Description

This command creates a property list in cot.xml.

4.4.22.2 Syntax

```
createFederationPropertyList(providerID, listName)
```

Argument	Definition
providerID	Specifies the provider ID.
listName	Specifies the name of the property list.

4.4.22.3 Example

The following command creates property list `mylistA`:

```
wls:/mydomain/serverConfig> createFederationPropertyList('providerA','mylistA')
```

4.4.23 createFederationPropertyListInMap

Online command that creates a property list nested in a property map.

4.4.23.1 Description

This command creates a property list, nested in a property map, in cot.xml.

4.4.23.2 Syntax

```
createFederationPropertyListInMap(providerID, mapName, listName)
```

Argument	Definition
providerID	Specifies the provider ID.
mapName	Specifies an existing property map to contain the nested list.
listName	Specifies the name of the property list.

4.4.23.3 Example

The following command creates nested property list mylistA:

```
wls:/mydomain/serverConfig>
createFederationPropertyListInMap('providerA','mymapA','mylistA')
```

4.4.24 createFederationPropertyMap

Online command that creates a property map.

4.4.24.1 Description

This command that creates a property map in cot.xml.

4.4.24.2 Syntax

```
createFederationPropertyMap(providerID, mapName)
```

Argument	Definition
providerID	Specifies the provider ID.
mapName	Specifies the name of the property map to be added to cot.xml.

4.4.24.3 Example

The following command creates property map `mymapA`:

```
wls:/mydomain/serverConfig> createFederationPropertyMap('providerA','mymapA')
```

4.4.25 createFederationPropertyMapInMap

Online command that creates a nested property map.

4.4.25.1 Description

This command that creates a property map, nested in another property map, in cot.xml.

4.4.25.2 Syntax

```
createFederationPropertyMapInMap(providerID, mapName, nestedMapName)
```

Argument	Definition
providerID	Specifies the provider ID.
mapName	Specifies the name of an existing property map.
nestedMapName	Specifies the name of the property map to be nested inside mapName in cot.xml.

4.4.25.3 Example

The following command creates nested property map `nestedmapA`:

```
wls:/mydomain/serverConfig>
createFederationPropertyMapInMap('providerA','mymapA','nestedmapA')
```

4.4.26 createPeerProviderEntry

Online command that creates a peer provider property map entry.

4.4.26.1 Description

This command creates a peer provider as a Map property entry to cot.xml.

4.4.26.2 Syntax

```
createPeerProviderEntry(providerID, description, providerType, version)
```

Argument	Definition
providerID	Specifies the provider ID to be created.
description	This is the description of the provider ID.
providerType	Specifies the provider type of the peer provider to be created.
version	Specifies the version of the peer provider to be created.

4.4.26.3 Example

The following command creates a SAML 2.0 service provider:

```
wls:/mydomain/serverConfig> createPeerProviderEntry('providerA','idp
test','SP','SAML2.0')
```

4.4.27 getConfigListValueInMap

Online command that returns a list nested in a map.

4.4.27.1 Description

This command returns a list, nested in a map, from config.xml.

4.4.27.2 Syntax

```
getConfigListValueInMap(configName, mapName, listName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be accessed.
mapName	Specifies the name of the property map.
listName	Specifies the name of the list to be fetched from the map.

4.4.27.3 Example

The following command returns `mylistA`:

```
wls:/mydomain/serverConfig>
getConfigListValueInMap('serverConfig','mymapA','mylistA')
```

4.4.28 getConfigMapEntryInMap

Online command that returns a map property entry nested in a map.

4.4.28.1 Description

This command returns a map property entry, nested in a map, from config.xml.

4.4.28.2 Syntax

```
getConfigMapEntryInMap(configName, mapname, nestedMapName, propName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be accessed.
mapName	Specifies the name of the property map.
nestedmapName	Specifies the name of the nested property map.
propName	Specifies the name of the property to be fetched from the nested map.

4.4.28.3 Example

The following command returns property entry `myvarA`:

```
wls:/mydomain/serverConfig>
getConfigMapEntryInMap('serverconfig','mymap','nestedmapA','myvarA')
```

4.4.29 getConfigProperty

Online command that returns a property value.

4.4.29.1 Description

This command returns a property value from config.xml.

4.4.29.2 Syntax

```
getConfigProperty(configName, propName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be accessed.
propName	Specifies the name of the property to be fetched from the nested map.

4.4.29.3 Example

The following command returns property `myvarA`:

```
wls:/mydomain/serverConfig> getConfigProperty('serverconfig','myvarA')
```

4.4.30 getConfigPropertyList

Online command that returns a property list.

4.4.30.1 Description

This command returns a property list from config.xml.

4.4.30.2 Syntax

```
getConfigPropertyList(configName, listName)
```

Argument	Definition
configName	Specifies the configuration name.
listName	Specifies the name of the property list to be fetched from config.xml.

4.4.30.3 Example

The following command returns mylistA:

```
wls:/mydomain/serverConfig> getConfigPropertyList('serverconfig','mylistA')
```

4.4.31 getConfigPropertyMapEntry

Online command that returns a property value from a map.

4.4.31.1 Description

This command returns a property value from a map in config.xml.

4.4.31.2 Syntax

```
getConfigPropertyMapEntry(configName, mapName, propName)
```

Argument	Definition
configName	Specifies the configuration name (for example, idpsaml20, serverconfig, spsaml20, ..).
mapName	Specifies the name of the property map.
propName	Specifies the name of the property to be fetched from the map in config.xml.

4.4.31.3 Example

The following command returns property `propA`:

```
wls:/mydomain/serverConfig> getConfigPropertyMapEntry('serverconfig','mapA',
'propA')
```

4.4.32 getFederationListValueInMap

Online command that returns a list value nested in a map.

4.4.32.1 Description

This command returns a list value nested in a map from cot.xml.

4.4.32.2 Syntax

```
getFederationListValueInMap(providerID, mapName, listName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map.
listName	Specifies the name of the list to be fetched from the map.

4.4.32.3 Example

The following command returns nested list mylistA:

```
wls:/mydomain/serverConfig>
getFederationListValueInMap('providerA','mymapA','mylistA')
```

4.4.33 getFederationMapEntryInMap

Online command that returns a map property entry nested in a map.

4.4.33.1 Description

This command returns a map property entry, nested in a map, from cot.xml.

4.4.33.2 Syntax

```
getFederationMapEntryInMap(providerID, mapname, nestedMapName, propName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map.
nestedmapName	Specifies the name of the nested property map.

Argument	Definition
propName	Specifies the name of the property to be fetched from the nested map.

4.4.33.3 Example

The following command returns property entry `myvarA`:

```
wls:/mydomain/serverConfig>
getFederationMapEntryInMap('providerA','mymap','nestedmapA','myvarA')
```

4.4.34 getFederationProperty

Online command that returns a property value.

4.4.34.1 Description

This command returns a property value from cot.xml.

4.4.34.2 Syntax

```
getFederationProperty(providerID, propName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
propName	Specifies the name of the property to be fetched from cot.xml.

4.4.34.3 Example

The following command returns property `myvarA`:

```
wls:/mydomain/serverConfig> getFederationProperty('providerA','myvarA')
```

4.4.35 getFederationPropertyList

Online command that returns a property list.

4.4.35.1 Description

This command returns a property list from cot.xml.

4.4.35.2 Syntax

```
getFederationPropertyList(providerID, listName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
listName	Specifies the name of the list to be fetched from the map.

4.4.35.3 Example

The following command returns list `mylistA`:

```
wls:/mydomain/serverConfig> getFederationPropertyList('providerA','mylistA')
```

4.4.36 getFederationPropertyMapEntry

Online command that returns a property value from a map.

4.4.36.1 Description

This command returns a property value from a map in cot.xml.

4.4.36.2 Syntax

```
getFederationPropertyMapEntry(providerID, mapName, propName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map.
propName	Specifies the name of the property to be fetched from the nested map.

4.4.36.3 Example

The following command returns property `propA` from a map:

```
wls:/mydomain/serverConfig> getFederationPropertyMapEntry('providerA','mapA',
'propA')
```

4.4.37 listCustomAuthnEngines

Online command that returns a list of custom authentication integration engines.

4.4.37.1 Description

This command returns a list of custom authentication integration engines from config.xml.

4.4.37.2 Syntax

```
listCustomAuthnEngines()
```

4.4.37.3 Example

The following command returns the list of all SP engines:

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

4.4.38 listCustomSPEngines

Online command that returns a list of custom SP integration engines.

4.4.38.1 Description

This command returns a list of custom service provider (SP) integration engines from config.xml.

4.4.38.2 Syntax

```
listCustomSPEngines()
```

4.4.38.3 Example

The following command returns the list of all SP integration engines:

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

4.4.39 loadMetadata

Online command that loads metadata from an input file.

4.4.39.1 Description

This command loads metadata from an input file into cot.xml.

4.4.39.2 Syntax

```
loadMetadata(metadatafile,description)
```

Argument	Definition
metadatafile	Specifies the metadata file of the peer provider to be added or updated.
description	This is a brief description of the peer provider to be loaded.

4.4.39.3 Example

The following command loads metadata from the file `metadatafile.xml`:

```
wls:/mydomain/serverConfig> loadMetadata('/home/metadatafile.xml','some
description')
```

4.4.40 removeConfigListInMap

Online command that removes a list property nested in a map.

4.4.40.1 Description

This command removes a list property nested in a map from config.xml.

4.4.40.2 Syntax

```
removeConfigListInMap(configName, mapName, listName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be accessed.
mapName	Specifies the name of the property map.
listName	Specifies the name of the list to be removed from the map.

4.4.40.3 Example

The following command removes the list property mylistA:

```
wls:/mydomain/serverConfig>
removeConfigListInMap('serverConfig','mymapA','mylistA')
```

4.4.41 removeConfigMapEntryInMap

Online command that removes a map property nested in a map.

4.4.41.1 Description

This command removes a map property entry nested in a map from config.xml.

4.4.41.2 Syntax

```
removeConfigMapEntryInMap(configName, mapname, nestedMapName, propName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be accessed.
mapName	Specifies the name of the property map.
nestedmapName	Specifies the name of the nested property map.
propName	Specifies the name of the property to be removed from the nested map.

4.4.41.3 Example

The following command removes the nested property `myvarA`:

```
wls:/mydomain/serverConfig>
removeConfigMapEntryInMap('serverconfig','mymap','nestedmapA','myvarA')
```

4.4.42 removeConfigMapInMap

Online command that removes a map property nested in a map.

4.4.42.1 Description

This command removes a map property entry nested in a map from config.xml.

4.4.42.2 Syntax

```
removeConfigMapEntryInMap(configName, mapname, nestedMapName, propName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
mapName	Specifies the name of the property map.
nestedmapName	Specifies the name of the nested property map.
propName	Specifies the name of the property to be removed from the nested map.

4.4.42.3 Example

The following command removes the nested property myvarA:

```
wls:/mydomain/serverConfig>
removeConfigMapEntryInMap('serverconfig','mymap','nestedmapA','myvarA')
```

4.4.43 removeConfigProperty

Online command that removes a configuration property.

4.4.43.1 Description

This command removes a property from config.xml.

4.4.43.2 Syntax

```
removeConfigProperty(configName, propName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
propName	Specifies the name of the property to be removed.

4.4.43.3 Example

The following command removes the property `myvarA`:

```
wls:/mydomain/serverConfig> removeConfigProperty('serverconfig','myvarA')
```

4.4.44 removeConfigPropertyList

Online command that removes a configuration property list.

4.4.44.1 Description

This command removes a property list from config.xml.

4.4.44.2 Syntax

```
removeConfigPropertyList(configName, listName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
listName	Specifies the name of the property list to be removed.

4.4.44.3 Example

The following command removes the property list `mylistA`:

```
wls:/mydomain/serverConfig> removeConfigPropertyList('serverconfig','mylistA')
```

4.4.45 removeConfigPropertyMap

Online command that removes a property map.

4.4.45.1 Description

This command removes a property map in config.xml.

4.4.45.2 Syntax

```
removeConfigPropertyMap(configName, mapName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
mapName	Specifies the name of the property map to be removed.

4.4.45.3 Example

The following command removes mapA:

```
wls:/mydomain/serverConfig> removeConfigPropertyMap('serverconfig','mapA')
```

4.4.46 removeConfigPropertyMapEntry

Online command that removes a property value from a map.

4.4.46.1 Description

This command removes a property value from a map in config.xml.

4.4.46.2 Syntax

```
removeConfigPropertyMapEntry(configName, mapName, propName)
```

Argument	Definition
configName	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
mapName	Specifies the name of the property map to be updated.
propName	Specifies the name of the property to be removed from the map.

4.4.46.3 Example

The following command removes property propA:

```
wls:/mydomain/serverConfig> removeConfigPropertyMapEntry('serverconfig','mapA',
'propA')
```

4.4.47 removeFederationListInMap

Online command that removes a property list in a map.

4.4.47.1 Description

This command removes a property list in a map, in cot.xml.

4.4.47.2 Syntax

```
removeFederationListInMap(providerID, mapName, listName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map.
listName	Specifies the name of the property list to be removed.

4.4.47.3 Example

The following command removes mylistA in mymapA:

```
wls:/mydomain/serverConfig>
removeFederationListInMap('providerA','mymapA','mylistA')
```

4.4.48 removeFederationMapInMap

Online command that removes a nested map in a map.

4.4.48.1 Description

This command removes a property map nested inside a map in cot.xml.

4.4.48.2 Syntax

```
removeFederationMapInMap(providerID, mapname, nestedMapName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map containing the nested map.
nestedmapName	Specifies the name of the nested property map to be removed.

4.4.48.3 Example

The following command removes `nestedmapA` in `mymap`:

```
wls:/mydomain/serverConfig>
removeFederationMapInMap('providerA','mymap','nestedmapA')
```

4.4.49 removeFederationMapEntryInMap

Online command that removes a nested map property entry in a map.

4.4.49.1 Description

This command removes a property name/value pair to a map nested inside a map in cot.xml.

4.4.49.2 Syntax

```
removeFederationMapEntryInMap(providerID, mapname, nestedMapName, propName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map containing the nested map.
nestedmapName	Specifies the name of the nested property map.
propName	Specifies the name of the property to be removed from the nested map.

4.4.49.3 Example

The following command removes map property entry myvarA:

```
wls:/mydomain/serverConfig>
removeFederationMapEntryInMap('providerA','mymap','nestedmapA','myvarA')
```

4.4.50 removeFederationProperty

Online command that removes a property value.

4.4.50.1 Description

This command removes a property entry in cot.xml.

4.4.50.2 Syntax

```
removeFederationProperty(providerID, propName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be updated.
propName	Specifies the name of the property to be removed.

4.4.50.3 Example

The following command removes the provider property myvarA:

```
wls:/mydomain/serverConfig> removeFederationProperty('providerA','myvarA')
```

4.4.51 removeFederationPropertyList

Online command that removes a property list entry.

4.4.51.1 Description

This command removes a property list entry in cot.xml.

4.4.51.2 Syntax

```
removeFederationPropertyList(providerID, listName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
listName	Specifies the name of the property list to be removed.

4.4.51.3 Example

The following command removes `mylistA`:

```
wls:/mydomain/serverConfig> removeFederationPropertyList('providerA','mylistA')
```

4.4.52 removeFederationPropertyMap

Online command that removes a property map.

4.4.52.1 Description

This command removes a property map in cot.xml.

4.4.52.2 Syntax

```
removeFederationPropertyMap(providerID, mapName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map to be removed.

4.4.52.3 Example

The following command removes a map:

```
wls:/mydomain/serverConfig> removeFederationPropertyMap('providerA','mapA')
```

4.4.53 removeFederationPropertyMapEntry

Online command that removes a property value from a map.

4.4.53.1 Description

This command removes a property value from a map in cot.xml.

4.4.53.2 Syntax

```
removeFederationPropertyMapEntry(providerID, mapName, propName)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be accessed.
mapName	Specifies the name of the property map to be updated.
propName	Specifies the name of the property to be removed from the map.

4.4.53.3 Example

The following command removes property `propA` from a map:

```
wls:/mydomain/serverConfig> removeFederationPropertyMapEntry('providerA','mapA',
'propA')
```

4.4.54 removePeerProviderEntry

Online command that removes a peer provider entry.

4.4.54.1 Description

This command removes a peer provider entry from cot.xml.

4.4.54.2 Syntax

```
removePeerProviderEntry(providerID)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be removed.

4.4.54.3 Example

The following command removes providerA:

```
wls:/mydomain/serverConfig> removePeerProviderEntry('providerA')
```

4.4.55 setConfigProperty

Online command that sets a property value in config.xml.

4.4.55.1 Description

This command adds or updates a property value in config.xml.

4.4.55.2 Syntax

```
setConfigProperty(configname, propName, value, type)
```

Argument	Definition
configname	Specifies the name of the configuration (for example, idpsaml20, serverconfig, spsaml20, ..) to be updated.
propName	Specifies the name of the property to be added/updated in config.xml.
value	Specifies the property value.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.55.3 Example

The following command sets the property `myvarA` and its value in the server configuration:

```
wls:/mydomain/serverConfig>
setConfigProperty('serverconfig','myvarA','myvalA','string')
```

4.4.56 setCustomAuthnEngine

Online command that updates a custom authentication integration engine.

4.4.56.1 Description

This command updates a custom authentication integration engine in config.xml.

4.4.56.2 Syntax

```
setCustomAuthnEngine(engineID, name, [enabled], [webContext], [authnRelativePath],
[logoutRelativePath], [logoutEnabled])
```

Argument	Definition
engineID	Specifies the engine ID of an existing engine.
name	Specifies the name of the custom engine.
enabled	This flag specifies whether the engine is enabled (true) or not (false).
webContext	Specifies the web context for the engine.
authnRelativePath	Specifies the authentication relative path URL for the engine.
logoutRelativePath	Specifies the logout relative path URL for the engine.
logoutEnabled	This flag is set true to enable logout for the engine, else false.

4.4.56.3 Example

The following command updates the configuration of custom authentication engine `abcdef`:

```
wls:/mydomain/serverConfig> setCustomAuthnEngine('abcdef',
'custom one','false','oracle:fed:authentication:unspecified','webcontext')
```

4.4.57 setCustomSPEngine

Online command that updates a custom SP integration engine.

4.4.57.1 Description

This command updates an existing custom SP integration engine in config.xml.

4.4.57.2 Syntax

```
setCustomSPEngine(engineID, name, [enabled, [authnMech], [webContext],
[authnRelativePath], [logoutRelativePath], [logoutEnabled])
```

Argument	Definition
engineID	Specifies the engine ID of an existing custom engine.
name	Specifies the name of the custom engine.
enabled	This flag specifies whether the engine is enabled (true) or not (false).
authnMech	Specifies the authentication mechanism for the engine.
webContext	Specifies the web context for the engine.
authnRelativePath	Specifies the authentication relative path URL for the engine.
logoutRelativePath	Specifies the logout relative path URL for the engine.
logoutEnabled	This flag is set true to enable logout for the engine, else false.

4.4.57.3 Example

The following command sets the name and the enabled flag for the engine with ID `engineID2`:

```
wls:/mydomain/serverConfig> setCustomSPEngine('engineid2','test','true')
```

4.4.58 setFederationProperty

Online command that adds or updates a property value.

4.4.58.1 Description

This command adds a property entry or updates an existing entry in cot.xml.

4.4.58.2 Syntax

```
setFederationProperty(providerID, propName, value, type)
```

Argument	Definition
providerID	Specifies the name of the peer provider to be updated.
propname	Specifies the name of the property to be added/updated in cot.xml.
value	Specifies the property value.
type	Specifies the type of property, BOOLEAN or STRING or LONG.

4.4.58.3 Example

The following command creates the property `myvarA` and sets its value:

```
wls:/mydomain/serverConfig>
setFederationProperty('providerA','myvarA','myvalA','string')
```

4.5 Directory Integration Platform Commands

Some of the Directory Integration Platform (DIP) tools use WLST internally, and therefore, there are no custom WLST commands available to run from the WLST command prompt or to use within scripts. For information on DIP tools, see "Directory

Integration Platform Tools" in the *Oracle Fusion Middleware User Reference for Oracle Identity Management*.

4.6 Security Commands

Use the WLST security commands listed in Table 4–5 to operate on a domain policy or credential store, and to migrate policies and credentials from a source repository to a target repository.

Table 4–5 WLST Security Commands

Use this command...	To...	Use with WLST...
createAppRole	Create a new application role.	Online
deleteAppRole	Remove an application role.	Online
grantAppRole	Add a principal to a role.	Online
revokeAppRole	Remove a principal from a role.	Online
listAppRoles	List all roles in an application.	Online
listAppRolesMembers	List all members in an application role.	Online
grantPermission	Create a new permission.	Online
revokePermission	Remove a permission.	Online
listPermissions	List all permissions granted to a principal.	Online
deleteAppPolicies	Remove all policies in an application.	Online
migrateSecurityStore	Migrate policies or credentials from a source repository to a target repository.	Offline
listCred	Obtain the list of attribute values of a credential.	Online
updateCred	Modify the attribute values of a credential.	Online
createCred	Create a new credential.	Online
deleteCred	Remove a credential.	Online
modifyBootStrapCredential	Update bootstrap credential store	Offline
reassociateSecurityStore	Reassociate policies and credentials to an LDAP repository	Online
upgradeSecurityStore	Upgrade security data from data used with release 10.1.x to data used with release 11.	Offline

4.6.1 createAppRole

Online command that creates a new application role.

4.6.1.1 Description

Creates a new application role in the domain policy store with a given application and role name. In the event of an error, the command returns a `WLSTException`.

4.6.1.2 Syntax

```
createAppRole(appStripe, appRoleName)
```

Argument	Definition
appStripe	Specifies an application stripe.
appRoleName	Specifies a role name.

4.6.1.3 Example

The following invocation creates a new application role with application stripe `myApp` and role name `myRole`:

```
wls:/mydomain/serverConfig> createAppRole(appStripe="myApp", appRoleName="myRole")
```

4.6.2 deleteAppRole

Online command that removes an application role.

4.6.2.1 Description

Removes an application role in the domain policy store with a given application and role name. In the event of an error, the command returns a `WLSTException`.

4.6.2.2 Syntax

```
createAppRole(appStripe, appRoleName)
```

Argument	Definition
appStripe	Specifies an application stripe.
appRoleName	Specifies a role name.

4.6.2.3 Example

The following invocation removes the role with application stripe `myApp` and role name `myRole`:

```
wls:/mydomain/serverConfig> createAppRole(appStripe="myApp", appRoleName="myRole")
```

4.6.3 grantAppRole

Online command that adds a principal to a role.

4.6.3.1 Description

Adds a principal (class or name) to a role with a given application stripe and name. In the event of an error, the command returns a `WLSTException`.

4.6.3.2 Syntax

```
grantAppRole(appStripe, appRoleName,principalClass, principalName)
```

Argument	Definition
appStripe	Specifies an application stripe.
appRoleName	Specifies a role name.
principalClass	Specifies the fully qualified name of a class.
principalName	Specifies the principal name.

4.6.3.3 Example

The following invocation adds a principal to the role with application stripe `myApp` and role name `myRole`:

```
wls:/mydomain/serverConfig> grantAppRole(appStripe="myApp",
appRoleName="myRole",principalClass="com.example.xyzPrincipal",
principalName="myPrincipal")
```

4.6.4 revokeAppRole

Online command that removes a principal from a role.

4.6.4.1 Description

Removes a principal (class or name) from a role with a given application stripe and name. In the event of an error, the command returns a `WLSTException`.

4.6.4.2 Syntax

```
revokeAppRole(appStripe, appRoleName, principalClass, principalName)
```

Argument	Definition
appStripe	Specifies an application stripe.
appRoleName	Specifies a role name.
principalClass	Specifies the fully qualified name of a class.
principalName	Specifies the principal name.

4.6.4.3 Example

The following invocation removes a principal to the role with application stripe `myApp` and role name `myRole`:

```
wls:/mydomain/serverConfig> revokeAppRole(appStripe="myApp",
appRoleName="myRole",principalClass="com.example.xyzPrincipal",
principalName="myPrincipal")
```

4.6.5 listAppRoles

Online command that lists all roles in an application.

4.6.5.1 Description

Lists all roles within a given application stripe. In the event of an error, the command returns a `WLSTException`.

4.6.5.2 Syntax

```
listAppRoles(appStripe)
```

Argument	Definition
appStripe	Specifies an application stripe.

4.6.5.3 Example

The following invocation returns all roles with application stripe `myApp`:

```
wls:/mydomain/serverConfig> listAppRoles(appStripe="myApp")
```

4.6.6 listAppRolesMembers

Online command that lists all members in a role.

4.6.6.1 Description

Lists all members in a role with a given application stripe and role name. In the event of an error, the command returns a `WLSTException`.

4.6.6.2 Syntax

```
listAppRoleMembers(appStripe, appRoleName)
```

Argument	Definition
appStripe	Specifies an application stripe.
appRoleName	Specifies a role name.

4.6.6.3 Example

The following invocation returns all members in the role with application stripe `myApp` and role name `myRole`:

```
wls:/mydomain/serverConfig> listAppRoleMembers(appStripe="myApp",
appRoleName="myRole")
```

4.6.7 grantPermission

Online command that creates a new permission.

4.6.7.1 Description

Creates a new permission for a given code base or URL. In the event of an error, the command returns a `WLSTException`.

4.6.7.2 Syntax

Optional arguments are enclosed in between square brackets.

```
grantPermission([appStripe,] [codeBaseURL,] [principalClass,] [principalName,]
permClass, [permTarget,] [permAction])
```

Argument	Definition
appStripe	Specifies an application stripe. If not specified, the command works on system policies.
codeBaseURL	Specifies the URL of the code granted the permission.
principalClass	Specifies the fully qualified name of a class (grantee).
principalName	Specifies the name of the grantee principal.
permClass	Specifies the fully qualified name of the permission class.
permTarget	Specifies, when available, the name of the permission target. Some permissions may not include this attribute.
permAction	Specifies a comma-separated list of actions granted. Some permissions may not include this attribute and the actions available depend on the permission class.

4.6.7.3 Examples

The following invocation creates a new application permission (for the application with application stripe `myApp`) with the specified data:

```
wls:/mydomain/serverConfig> grantPermission(appStripe="myApp",
principalClass="my.custom.Principal",  principalName="manager",
permClass="java.security.AllPermission")
```

The following invocation creates a new system permission with the specified data:

```
wls:/mydomain/serverConfig> grantPermission(principalClass="my.custom.Principal",
principalName="manager",  permClass="java.io.FilePermission",
permTarget="/tmp/fileName.ext", permTarget="/tmp/fileName.ext",
permAction="read,write")
```

4.6.8 revokePermission

Online command that removes a permission.

4.6.8.1 Description

Removes a permission for a given code base or URL. In the event of an error, the command returns a `WLSTException`.

4.6.8.2 Syntax

Optional arguments are enclosed in between square brackets.

```
revokePermission([appStripe,] [codeBaseURL,] [principalClass,] [principalName,]
permClass, [permTarget,] [permAction])
```

Argument	Definition
appStripe	Specifies an application stripe. If not specified, the command works on system policies.
codeBaseURL	Specifies the URL of the code granted the permission.
principalClass	Specifies the fully qualified name of a class (grantee).
principalName	Specifies the name of the grantee principal.
permClass	Specifies the fully qualified name of the permission class.
permTarget	Specifies, when available, the name of the permission target. Some permissions may not include this attribute.
permAction	Specifies a comma-separated list of actions granted. Some permissions may not include this attribute and the actions available depend on the permission class.

4.6.8.3 Examples

The following invocation removes the application permission (for the application with application stripe `myApp`) with the specified data:

```
wls:/mydomain/serverConfig> revokePermission(appStripe="myApp",
principalClass="my.custom.Principal", principalName="manager",
permClass="java.security.AllPermission")
```

The following invocation removes the system permission with the specified data:

```
wls:/mydomain/serverConfig> revokePermission(principalClass="my.custom.Principal",
principalName="manager",  permClass="java.io.FilePermission",
```

```
permTarget="/tmp/fileName.ext", permAction="read,write")
```

4.6.9 listPermissions

Online command that lists all permissions granted to a given principal.

4.6.9.1 Description

Lists all permissions granted to a given principal. In the event of an error, the command returns a `WLSTException`.

4.6.9.2 Syntax

Optional arguments are enclosed in between square brackets.

```
listPermissions([appStripe,] principalClass, principalName)
```

Argument	Definition
appStripe	Specifies an application stripe. If not specified, the command works on system policies.
principalClass	Specifies the fully qualified name of a class (grantee).
principalName	Specifies the name of the grantee principal.

4.6.9.3 Examples

The following invocation lists all permissions granted to a principal by the policies of application `myApp`:

```
wls:/mydomain/serverConfig> listPermissions(appStripe="myApp",
principalClass="my.custom.Principal",principalName="manager")
```

The following invocation lists all permissions granted to a principal by system policies:

```
wls:/mydomain/serverConfig> listPermissions(principalClass="my.custom.Principal",
principalName="manager")
```

4.6.10 deleteAppPolicies

Online command that removes all policies with a given application stripe.

4.6.10.1 Description

Removes all policies with a given application stripe. In the event of an error, the command returns a `WLSTException`.

4.6.10.2 Syntax

```
deleteAppPolicies(appStripe)
```

Argument	Definition
appStripe	Specifies an application stripe. If not specified, the command works on system policies.

4.6.10.3 Example

The following invocation removes all policies of application `myApp`:

```
wls:/mydomain/serverConfig> deleteAppPolicies(appStripe="myApp")
```

4.6.11 migrateSecurityStore

Offline command that migrates identities, application-specific, system policies, a specific credential folder, or all credentials.

4.6.11.1 Description

Migrates identities, application-specific, or system policies from a source repository to a target repository. Migrates a specific credential folder or all credentials.

The kinds of the repositories where the source and target data is stored is transparent to the command, and any combination of file-based and LDAP-based repositories is allowed (LDAP-repositories must use an OVD or an OID LDAP server only). In the event of an error, the command returns a `WLSTException`.

4.6.11.2 Syntax

The command syntax varies depending on the scope (system or application-specific or both) of the policies being migrated.

Optional arguments are enclosed in square brackets.

To migrate identities, use the following syntax:

```
migrateSecurityStore(type="idStore", configFile, src, dst, [dstLdifFile])
```

To migrate all policies (system *and* application-specific, for all applications) use the following syntax

```
migrateSecurityStore(type="policyStore", configFile, src, dst,[overWrite])
```

To migrate *just* system policies, use the following syntax:

```
migrateSecurityStore(type="globalPolicies", configFile, src, dst, [overWrite])
```

To migrate *just* application-specific policies, for one application, use the following syntax:

```
migrateSecurityStore(type="appPolicies", configFile,src, dst, srcApp, [dstApp,]
[overWrite,] [migrateIdStoreMapping])
```

To migrate *all* credentials, use the following syntax:

```
migrateSecurityStore(type="credStore", configFile, src, dst, [overWrite])
```

To migrate *just* one credential folder, use the following syntax:

```
migrateSecurityStore(type="folderCred", configFile,src, dst, [srcFolder,]
[dstFolde,] [srcConfigFile,] [overWrite])
```

Argument	Definition
type	Specifies the type of policies migrates. To migrate identities, set it to `idStore`. To migrate all policies (system and application-specific, for all applications), set to `policyStore`. To migrate just system policies, set to `globalPolicies`. To migrate just application-specific policies, set to `appPolicies`. To migrate all credentials, set to `credStore`. To migrate just one credential folder, set to `folderCred`.
configFile	Specifies the location of a configuration file `jps-config.xml` relative to the directory where the command is run. The configuration file passed need not be an actual domain configuration file, but it can be assembled *just* to specify the source and destination repositories of the migration.
src	Specifies the name of a jps-context in the configuration file passed to the argument `configFile`, where the source store is specified.
dst	Specifies the name of another jps-context in the configuration file passed to the argument `configFile`, where the destination store is specified.
srcApp	Specifies the name of the source application, that is, the application whose policies are being migrated.
dstApp	Specifies the name of the target application, that is, the application whose policies are being written. If unspecified, it defaults to the name of the source application.
srcFolder	Specifies the name of the folder from where credentials are migrated. This argument is optional. If unspecified, the credential store is assumed to have only one folder and the value of this argument defaults to the name of that folder.
dstFolder	Specifies the folder to where the source credentials are migrated. This argument is optional and, if unspecified, defaults to the folder passed to `srcFolder`.
srcConfigFile	Specifies the location of an alternate configuration file, and it is used in the special case in which credentials are not configured in the file passed to `configFile`. This argument is optional. If unspecified, it defaults to the value passed to `configFile`; if specified, the value passed to `configFile` is ignored.
overWrite	Specifies whether data in the target matching data being migrated should be overwritten by or merged with the source data. Optional and false by default. Set to true to overwrite matching data; set to false to merge matching data.
migrateIdStoreMapping	Specifies whether the migration of application policies should include or exclude the migration of enterprise policies. Optional and true by default. Set it to False to exclude enterprise policies from the migration of application policies.
dstLdifFile	Specifies the location where the LDIF file will be created. Required only if destination is an LDAP-based identity store. Notice that the LDIF file is not imported into the LDAP server; the importing of the file LDIF should be done manually, after the file has been edited to account for the appropriate attributes required in your LDAP server.

Note the following requirements about the passed arguments:

- The file `jps-config.xml` is found in the passed location.

- The file `jps-config.xml` includes the passed jps-contexts.
- The source and the destination context names are distinct. From these two contexts, the command determines the locations of the source and the target repositories involved in the migration.

4.6.11.3 Examples

The following invocation illustrates the migration of the file-based policies of application `PolicyServlet1` to file-based policies of application `PolicyServlet2`:

```
wls:/mydomain/serverConfig> migrateSecurityStore(type="appPolicies",
configFile="jps-congif.xml", src="default1", dst="context2",
srcApp="PolicyServlet1", dstApp="PolicyServlet2", overWrite="true")
```

The above invocation assumes that:

- The file `jps-config.xml` is located in the directory where the command is run (current directory).
- That file includes the following elements:

```
<serviceInstance name="policystore1.xml" provider="some.provider">
  <property name="location" value="jazn-data1.xml"/>
</serviceInstance>
<serviceInstance name="policystore2.xml" provider="some.provider">
  <property name="location" value="jazn-data2.xml"/>
</serviceInstance>
...
<jpsContext name="default1">
  <serviceInstanceRef ref="policystore1.xml"/>
  ...
</jpsContext>
<jpsContext name="context2">
  <serviceInstanceRef ref="policystore2.xml"/>
  ...
</jpsContext>
```

The file-based policies for the two applications involved in the migration are defined in the files `jazn-data1.xml` and `jazn-data2.xml`, which are not shown but assumed located in the current directory.

The following invocation illustrates the migration of file-based credentials from one location to another:

```
wls:/mydomain/serverConfig> migrateSecurityStore(type="credStore",
configFile="jps-congif.xml", src="default1", dst="context2")
```

The above invocation assumes that:

- The file `jps-config.xml` is located in the directory where the command is run (current directory).
- That file includes the following elements:

```
<serviceInstance name="credstore1" provider="some.provider">
  <property name="location" value="./credstore1/cwallet.sso"/>
</serviceInstance>
<serviceInstance name="credstore2" provider="some.provider">
  <property name="location" value="./credstore2/cwallet.sso"/>
</serviceInstance>
...
<jpsContext name="default1">
```

```
  <serviceInstanceRef ref="credstore1"/>
  ...
</jpsContext>
<jpsContext name="context2">
  <serviceInstanceRef ref="credstore2"/>
  ...
</jpsContext>
```

4.6.12 listCred

Online command that returns the list of attribute values of a credential in the domain credential store.

4.6.12.1 Description

Returns the list of attribute values of a credential in the domain credential store with given map name and key name. This command lists the data encapsulated in credentials of type password only. In the event of an error, the command returns a `WLSTException`.

4.6.12.2 Syntax

```
listCred(map, key)
```

Argument	Definition
map	Specifies a map name (folder).
key	Specifies a key name.

4.6.12.3 Example

The following invocation returns all the information (such as user name, password, URL, port, and description) in the credential with map name `myMap` and key name `myKey`:

```
wls:/mydomain/serverConfig> listCred(map="myMap", key="myKey")
```

4.6.13 updateCred

Online command that modifies the type, user name, and password of a credential.

4.6.13.1 Description

Modifies the type, user name, password, URL, and port number of a credential in the domain credential store with given map name and key name. This command can update the data encapsulated in credentials of type password only. In the event of an error, the command returns a `WLSTException`. This command runs in interactive mode only.

4.6.13.2 Syntax

Optional arguments are enclosed in square brackets.

```
updateCred(map, key, user, password, [desc])
```

Argument	Definition
map	Specifies a map name (folder).
key	Specifies a key name.

Argument	Definition
user	Specifies the credential user name.
password	Specifies the credential password.
desc	Specifies a string describing the credential.

4.6.13.3 Examples

The following invocation updates a password credential with the specified data:

```
wls:/mydomain/serverConfig> updateCred(map="myMap", key="myKey", user="myUsr",
password="myPassw", desc="updated passw cred to connect to app xyz")
```

4.6.14 createCred

Online command that creates a new credential in the domain credential store.

4.6.14.1 Description

Creates a new credential in the domain credential store with a given map name, key name, type, user name and password, URL and port number. In the event of an error, the command returns a `WLSTException`. This command runs in interactive mode only.

4.6.14.2 Syntax

Optional arguments are enclosed in square brackets.

```
createCred(map, key, user, password, [desc])
```

Argument	Definition
map	Specifies a map name (folder).
key	Specifies a key name.
user	Specifies the credential user name.
password	Specifies the credential password.
desc	Specifies a string describing the credential.

4.6.14.3 Examples

The following invocation creates a new password credential with the specified data:

```
wls:/mydomain/serverConfig> createCred(map="myMap, key="myKey", user="myUsr",
password="myPassw", desc="updated usr name and passw to connect to app xyz")
```

4.6.15 deleteCred

Online command that removes a credential in the domain credential store.

4.6.15.1 Description

Removes a credential with given map name and key name from the domain credential store. In the event of an error, the command returns a `WLSTException`.

4.6.15.2 Syntax

```
deleteCred(map,key)
```

Argument	Definition
map	Specifies a map name (folder).
key	Specifies a key name.

4.6.15.3 Examples

The following invocation removes the credential with map name `myMap` and key name `myKey`:

```
wls:/mydomain/serverConfig> deleteCred(map="myApp",key="myKey")
```

4.6.16 modifyBootStrapCredential

Offline command that updates a bootstrap credential store.

4.6.16.1 Description

Updates a bootstrap credential store with given user name and password. In the event of an error, the command returns a `WLSTException`.

Typically used in the following scenario: suppose that the domain policy and credential stores are LDAP-based, and the credentials to access the LDAP store (stored in the LDAP server) are changed. Then this command can be used to seed those changes into the bootstrap credential store.

4.6.16.2 Syntax

```
modifyBootStrapCredential(jpsConfigFile, username, password)
```

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run.
username	Specifies the distinguished name of the user in the LDAP store.
password	Specifies the password of the user.

4.6.16.3 Examples

Suppose that in the LDAP store, the password of the user with distinguished name `cn=orcladmin` has been changed to `welcome1`, and that the configuration file `jps-config.xml` is located in the current directory.

Then the following invocation changes the password in the bootstrap credential store to `welcome1`:

```
wls:/mydomain/serverConfig> modifyBootStrapCredential(jpsConfigFile=
'./jps-config.xml', username='cn=orcladmin', password='welcome1')
```

Any output regarding the audit service can be disregarded.

4.6.17 reassociateSecurityStore

Online command that migrates the policy and credential stores to an LDAP repository.

4.6.17.1 Description

Migrates, within a give domain, *both* the policy store and the credential store to a target LDAP server repository. The only kinds of LDAP servers allowed are OID or

OVD. In the event of an error, the command returns a `WLSTException`. This command runs in interactive mode only.

4.6.17.2 Syntax

```
reassociateSecurityStore(domain, admin, password, ldapurl, servertype, jpsroot)
```

Argument	Definition
domain	Specifies the domain name where the reassociating takes place.
admin	Specifies the administrator's user name on the LDAP server. The format is `cn=usrName`.
password	Specifies the password associated with the user specified for the argument `admin`.
ldapurl	Specifies the URI of the LDAP server. The format is `ldap//:host:port`, if you are using a default port, or `ldaps://host:port`, if you are using a secure LDAP port. The secure port must be configured specially for this function and it is distinct from the default (non-secure) port.
servertype	Specifies the kind of the target LDAP server. The only valid types are OID or OVD.
jpsroot	Specifies the root node in the target LDAP repository under which all data is migrated. The format is `cn=nodeName`.

4.6.17.3 Example

The following invocation reassociates the domain policies and credentials to an LDAP Oracle Internet Directory server:

```
wls:/mydomain/serverConfig> reassociateSecurityStore(domain="myDomain",
admin="cn=adminName", password="myPass", ldapurl="ldap://myhost.example.com:3060",
servertype="OID", jpsroot="cn=testNode")
```

4.6.18 upgradeSecurityStore

Offline command that migrates release 10.1.x security data to release 11 security data.

4.6.18.1 Description

Migrates identity, policy, and credential data used in release 10.1.x to security data that can be used with release 11. The migration of each kind of data is performed with separate invocations of this command. In the event of an error, the command returns a `WLSTException`.

4.6.18.2 Syntax

The syntax varies according to the type of data being updated.

To upgrade 10.1.x XML identity data to 11 XML identity data, use the following syntax:

```
updateSecurityStore(type="xmlIdStore", jpsConfigFile, srcJaznDataFile, srcRealm,
dst)
```

To upgrade a 10.1.x XML policy data to 11 XML policy data, use the following syntax:

```
updateSecurityStore(type="xmlPolicyStore", jpsConfigFile, srcJaznDataFile, dst)
```

To upgrade a 10.1.x OID LDAP-based policy data to 11 XML policy data, use the following syntax:

```
updateSecurityStore(type="oidPolicyStore", jpsConfigFile, srcJaznDataFile, dst)
```

To upgrade a 10.1.x XML credential data to 11 XML credential data, use the following syntax:

```
updateSecurityStore(type="xmlCredStore", jpsConfigFile, srcJaznDataFile, users,
dst)
```

Argument	Definition
type	Specifies the kind of security data being upgraded. The only valid values are xmlIdStore, xmlPolicyStore, oidPolicyStore, and xmlCredStore.
jpsConfigFile	Specifies the location of a configuration file `jps-config.xml` relative to the directory where the command is run. The target store of the upgrading is read from the context specified with the argument `dst`.
srcJaznDataFile	Specifies the location of a 10.1.x jazn data file relative to the directory where the command is run. This argument is required if the specified `type` is xmlIdStore, xmlPolicyStore, or xmlCredStore.
srcJaznConfigFile	Specifies the location of a 10.1.x jazn configuration file relative to the directory where the command is run. This argument is required if the specified `type` is oidPolicyStore.
srcRealm	Specifies the name of the realm from which identities need be migrated. This argument is required if the specified `type` is xmlIdStore.
users	Specifies a comma-separated list of users each formatted as *realmName/userName*. This argument is required if the specified `type` is xmlCredStore.
dst	Specifies the name of the jpsContext in the file passed to the argument jpsConfigFile where the destination store is configured. Optional. If unspecified, it defaults to the default context in the file passed in the argument jpsConfigFile.

4.6.18.3 Examples

The following invocation migrates 10.1.3 file-based identities to an 11 file-based identity store:

```
wls:/mydomain/serverConfig> upgradeSecurityStore(type="xmlIdStore",
jpsConfigFile="jps-config.xml", srcJaznDataFile="jazn-data.xml",
srcRealm="jazn.com")
```

The following invocation migrates a 10.1.3 OID-based policy store to an 11 file-based policy store:

```
wls:/mydomain/serverConfig> upgradeSecurityStore(type="oidPolicyStore",
jpsConfigFile="jps-config.xml", srcJaznDataFile="jazn-data.xml",
dst="destinationContext)
```

5

Oracle WebCenter Custom WLST Commands

This chapter describes WebLogic Scripting Tool (WLST) commands for Oracle WebCenter. These commands enable you to configure WebCenter applications from the command-line. For additional details about WebCenter application configuration, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Notes: To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Most configuration changes made using WebCenter WLST commands are only effective after you restart the Managed Server on which the WebCenter application is deployed. The only exceptions are the External Applications, Producer, and Import and Export WLST commands.

WebCenter WLST commands are described in the following sections:

5.1 Oracle WebCenter WSLT Command Categories

Oracle WebCenter WLST commands are grouped into the following categories (Table 5–1).

Most configuration changes made using WebCenter WLST commands are only effective after you restart the Managed Server on which the WebCenter application is deployed. The only exceptions are the External Applications, Producer, and Import and Export WLST commands.

Table 5–1 WLST Command Categories

Command Category	Description
Section 5.2, "General"	Manage WebCenter connections.
Section 5.3, "Content Repository"	Manage content repository connections and configure the Documents service.
Section 5.4, "Discussions and Announcements"	Manage discussions server connections.
Section 5.5, "External Applications"	Manage external application connections.
Section 5.6, "Instant Messaging and Presence"	Manage instant messaging and presence server connections.
Section 5.7, "Mail"	Manage mail server connections.
Section 5.8, "Producer"	Manage portlet producers.
Section 5.9, "RSS"	Manage proxy settings for the RSS service in WebCenter Spaces.
Section 5.10, "Search"	Manage Oracle Secure Enterprise Search (SES) connections.
Section 5.11, "Worklists"	Manage BPEL server connections.
Section 5.12, "WebCenter Spaces Workflows"	Manage the BPEL connection for WebCenter Spaces workflows.
Section 5.13, "Import and Export"	Export and import WebCenter Spaces applications, group spaces, group space templates, and producer metadata.

5.2 General

Use the General commands, listed in Table 5–2, to manage WebCenter connections.

Configuration changes made using these WebCenter WLST commands are only effective after restarting the Managed Server on which the WebCenter application is deployed. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 5–2 General WLST Commands

Use This Command...	To...	Use with WLST...
deleteConnection	Delete any WebCenter connection.	Online

5.2.1 deleteConnection

Module: Oracle WebCenter

Use with WLST: Online

5.2.1.1 Description

Deletes a named WebCenter connection.

If you use `deleteConnection` to delete a WSRP or PDK-Java producer connection (instead of using `deregisterWSRPProducer` or deregisterPDKJavaProducer), unused secondary connections will remain, which you might want to remove. For example, when you delete a WSRP producer connection, its associated Web Service connection remains; when you delete a PDK-Java producer connection, its associated URL connection remains.

5.2.1.2 Syntax

```
deleteConnection(appName, name, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.2.1.3 Example

The following example deletes a WebCenter connection.

```
wls:/weblogic/serverConfig> deleteConnection(appName='webcenter',
name='MyConnection')
```

5.3 Content Repository

Use the commands listed in Table 5–3 to manage content repository connections and configure the Documents service for a WebCenter application.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 5–3 Content Repository WLST Commands

Use this command...	To...	Use with WLST...
createJCRContentServerConnection	Create an Oracle Content Server repository connection.	Online
setJCRContentServerConnection	Edit an existing Oracle Content Server connection.	Online
listJCRContentServerConnections	List individual or all Oracle Content Server connections that are configured for a WebCenter application.	Online
createJCRPortalConnection	Create an Oracle Portal repository connection.	Online

Table 5–3 (Cont.) Content Repository WLST Commands

Use this command...	To...	Use with WLST...
setJCRPortalConnection	Edit an existing Oracle Portal repository connection.	Online
listJCRPortalConnections	List all Oracle Portal connections that are configured for a WebCenter application.	Online
createJCRFileSystemConnection	Create a connection to a file system.	Online
setJCRFileSystemConnection	Edit an existing file system repository connection.	Online
listJCRFileSystemConnections	List individual or all file system connections configured for a WebCenter application.	Online
listDocumentsSpacesProperties	List properties for the back-end Oracle Content Server repository that is being used by WebCenter Spaces.	Online
setDocumentsSpacesProperties	Modify properties for the back-end Oracle Content Server repository used by WebCenter Spaces.	Online
deleteDocumentsSpacesProperties	Delete properties for the back-end Oracle Content Server repository used by WebCenter Spaces.	Online

5.3.1 createJCRContentServerConnection

Module: Oracle WebCenter

Use with WLST: Online

5.3.1.1 Description

Creates an Oracle Content Server repository connection, for a named WebCenter application.

5.3.1.2 Syntax

```
createJCRContentServerConnection(appName, name, socketType, [url, serverHost,
serverPort, keystoreLocation, keystorePassword, privateKeyAlias,
privateKeyPassword, extAppId, isPrimary, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.

Argument	Definition
socketType	Specifies whether the Oracle Content Server connects on the content server listener port or the Web server filter, and whether the listener port is SSL enabled. Valid values are `socket`, `web`, and `socketssl`. This option has no default. Choose from: ■ **socket** - Use an `intradoc` socket connection to connect to the Oracle Content Server. The client IP address must be added to the list of authorized addresses in the Oracle Content Server. In this case, the client is the machine on which Oracle WebCenter is running. ■ **socketssl** - Use an `intradoc` socket connection to connect to the Oracle Content Server that is secured using the SSL protocol. The client's certificates must be imported in the server's trust store for the connection to be allowed. Because this is the most secure option, this is the recommended option whenever identity propagation is required (for example, in WebCenter Spaces). ■ **web** - Use an HTTP(S) connection to connect to the Oracle Content Server. Note that for WebCenter Spaces, this option is not suitable for the back-end Oracle Content Server repository that is being used to store group space and personal space documents, because it does not allow identity propagation.
url	Optional. Oracle Content Server URL. Required only if `socketType` is set to `web`. URL should be in the format: `http://<hostname>:<port>/<web root>/<plugin root>` For example, `http://mycontentserver/cms/idcplg`.
serverHost	Optional. Host name of the machine where the Oracle Content Server is running. Required if `socketType` is set to `socket` or `socketssl`.
serverPort	Optional. Port on which the Oracle Content Server listens. Required if `socketType` is set to `socket` or `socketssl`: ■ Socket - Port specified for the `incoming` provider in the server. ■ Socket SSL - Port specified for the `sslincoming` provider in the server. For example, `4444`
keystoreLocation	Optional. Location of key store that contains the private key used to sign the security assertions. Required only if `socketType` is set to `socketssl`. The key store location must be an absolute path.
keystorePassword	Optional. Password required to access the key store. Required only if `socketType` is set to `socketssl`.
privateKeyAlias	Optional. Client private key alias in the key store. The key is used to sign messages to the server. The public key corresponding to this private key must be imported in the server keystore. Required only if `socketType` is set to `socketssl`. The value for this argument must be a string that contains neither special characters nor white space.
privateKeyPassword	Optional. Password to be used with the private key alias in the key store. Required only if `socketType` is set to `socketssl`.

Argument	Definition
extAppId	Optional. External application used to authenticate WebCenter users against the Oracle Content Server. This value should match the name of an existing external application connection. See also listExtAppConnections. If `extAppId` is not set, no change is made to the authentication method or external application ID. If `extAppId` is set to an empty string, the authentication method used is `IDENTITY_PROPAGATION`. With this method, the WebCenter application and Oracle Content Server use the same identity store to authenticate users. Note that `extAppID` is mandatory when `socketType` is set to `web`.
isPrimary	Optional. Valid string values are `true` and `false` (note that single quotes should surround `true` and `false`). `true` specifies that this connection is the primary connection used by the Documents service. This argument defaults to `false`. In WebCenter Spaces, the primary connection is used to store group space folders and personal space folders.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.1.3 Examples

The following example creates a socket-based connection to an Oracle Content Server running on `myhost.com` at port `4444`. For authentication purposes, an existing external application named `myExtApp` is used. See also, createExtAppConnection.

```
wls:/weblogic/serverConfig> createJCRContentServerConnection(appName='webcenter',
name='myContentServerConnection', socketType='socket',
serverHost='myhost.com', serverPort='4444', extAppId='myExtApp',
isPrimary='true')
```

The following example creates an SSL socket-based connection to an Oracle Content Server repository.

```
wls:/weblogic/serverConfig> createJCRContentServerConnection(appName='webcenter',
name='myContentServerConnection', socketType='socketssl',
serverHost='myhost.com', serverPort='4444',
keystoreLocation='d:/scratch/keystore.xyz', keystorePassword='AlphaSquad7',
privateKeyAlias='enigma', privateKeyPassword='S0larPl3x1s',
extAppId='myExtApp')
```

5.3.2 setJCRContentServerConnection

Module: Oracle WebCenter

Use with WLST: Online

5.3.2.1 Description

Edits an existing Oracle Content Server connection. This command requires that you specify values for `appName` and `name`, plus one additional argument.

5.3.2.2 Syntax

```
setJCRContentServerConnection(appName, name, [socketType, url, serverHost,
```

```
serverPort, keystoreLocation, keystorePassword, privateKeyAlias,
privateKeyPassword, extAppId, isPrimary, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing Oracle Content Server connection.
socketType	Optional. Specifies whether the Oracle Content Server connects on the content server listener port or the Web server filter, and whether the listener port is SSL enabled. Valid values are `socket`, `web`, and `socketssl`. This option has no default. Choose from: ■ **socket** - Use an `intradoc` socket connection to connect to the Oracle Content Server. The client IP address must be added to the list of authorized addresses in the Oracle Content Server. In this case, the client is the machine on which Oracle WebCenter is running. ■ **socketssl** - Use an `intradoc` socket connection to connect to the Oracle Content Server that is secured using the SSL protocol. The client's certificates must be imported in the server's trust store for the connection to be allowed. Because this is the most secure option, this is the recommended option whenever identity propagation is required (for example, in WebCenter Spaces). ■ **web** - Use an HTTP(S) connection to connect to the Oracle Content Server. Note that for WebCenter Spaces, this option is not suitable for the back-end Oracle Content Server repository that is being used to store group space and personal space documents, because it does not allow identity propagation.
url	Optional. Oracle Content Server URL. Required only if `socketType` is set to `web`. URL should be in the format: `http://<hostname>:<port>/<web root>/<plugin root>` For example, `http://mycontentserver/cms/idcplg`.
serverHost	Optional. Host name of the machine where the Oracle Content Server is running. Required if `socketType` is set to `socket` or `socketssl`.
serverPort	Optional. Port on which the Oracle Content Server listens. Required if `socketType` is set to `socket` or `socketssl`: ■ Socket - Port specified for the `incoming` provider in the server. ■ Socket SSL - Port specified for the `sslincoming` provider in the server. For example, `4444`
keystoreLocation	Optional. Location of key store that contains the private key used to sign the security assertions. Required only if `socketType` is set to `socketssl`. The key store location must be an absolute path.
keystorePassword	Optional. Password required to access the key store. Required only if `socketType` is set to `socketssl`.
privateKeyAlias	Optional. Client private key alias in the key store. Required only if `socketType` is set to `socketssl`. The value for this argument must be a string that contains neither special characters nor white space.
privateKeyPassword	Optional. Password to be used with the private key alias in the key store. Required only if `socketType` is set to `socketssl`.

Argument	Definition
extAppId	Optional. External application used to authenticate WebCenter users against the Oracle Content Server. This value should match the name of an existing external application connection. See also listExtAppConnections. If `extAppId` is not set, no change is made to the authentication method or external application ID. If `extAppId` is set to an empty string, the authentication method used is `IDENTITY_PROPAGATION`. With this method, the WebCenter application and Oracle Content Server use the same identity store to authenticate users.
isPrimary	Optional. Valid string values are `true` and `false` (note that single quotes should surround `true` and `false`). `true` specifies that this connection is the primary connection used by the Documents service. This argument defaults to `false`. In WebCenter Spaces, the primary connection is used to store group space folders and personal space folders.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.2.3 Examples

The following example edits connection details for an Oracle Content Server connection.

```
wls:/weblogic/serverConfig>setJCRContentServerConnection(appName='webcenter',
name='myContentServerConnection', socketType='socket',
serverHost='myhost.com',  serverPort='4444',
extAppId='myExtApp', isPrimary='true')
```

The following example edits connection details for an Oracle Content Server connection.

```
wls:/weblogic/serverConfig>setJCRContentServerConnection(appName='webcenter',
name='myContentServerConnection', socketType='socketssl',
serverHost='myhost.com', serverPort='8443',
keystoreLocation='d:/keys/here', keystorePassword='T0PS3CR3T',
privateKeyAlias='TekJansen', privateKeyPassword='LadyNocturne',
extAppId='myExtApp', isPrimary='true')
```

5.3.3 listJCRContentServerConnections

Module: Oracle WebCenter

Use with WLST: Online

5.3.3.1 Description

Without any arguments, this command lists all of the Oracle Content Server connections that are configured for a named WebCenter application.

5.3.3.2 Syntax

```
listJCRContentServerConnections(appName, [verbose],
[name, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
verbose	Optional. Displays content repository connection details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listJCRContentServerConnections` lists all Oracle Content Server connections that are configured for a WebCenter application, along with their details. When set to `false`, only connection names are listed. This argument defaults to `false`.
name	Optional. Name of an existing Oracle Content Server connection. When specified you can view connection details for a specific Oracle Content Server connection. If you supply a value for `name`, you must supply a value for `verbose`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.3.3 Examples

The following example lists Oracle Content Server connections configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig> listJCRContentServerConnections(appName='webcenter')
```

The following example lists all properties of the Oracle Content Server connection named `myContentServerConnection1`. The connection named `myContentServerConnection1` must exist and be an Oracle Content Server connection. If, for example, you specify an Oracle Portal connection, the properties are not listed and an error is displayed.

```
wls:/weblogic/serverConfig>listJCRContentServerConnections(appName='webcenter',
verbose=true, name='myContentServerConnection1')
```

5.3.4 createJCRPortalConnection

Module: Oracle WebCenter

Use with WLST: Online

5.3.4.1 Description

Creates an Oracle Portal repository connection.

5.3.4.2 Syntax

```
createJCRPortalConnection(appName, name, dataSource, [extAppId, isPrimary,
server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.

Argument	Definition
dataSource	JNDI DataSource location used to connect to the portal. For example: `jdbc/MyPortalDS` The datasource must be on the server where the WebCenter application is deployed.
extAppId	Optional. External application used to authenticate WebCenter users against Oracle Portal. This value should match the name of an existing external application connection. See also listExtAppConnections. If `extAppId` is not set, no change is made to the authentication method or external application ID. If `extAppId` is set to an empty string, the authentication method used is `IDENTITY_PROPAGATION`. With this method, the WebCenter application and Oracle Portal use the same identity store to authenticate users.
isPrimary	Optional. Valid string values are `true` and `false`. `true` specifies that this connection is the primary connection used by the Documents service. This argument defaults to `false`. In WebCenter Spaces, the primary connection must be an Oracle Content Server connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.4.3 Example

The following example creates a Oracle Portal connection named `myPortalConnection` using the data source `jdbc/portalDS` and specifies that an external application, named `myExtApp`, is used for authentication.

```
wls:/weblogic/serverConfig> createJCRPortalConnection(appName='myApp',
name='myPortalConnection', dataSource='jdbc/portalDS', extAppId='myExtApp',
isPrimary='true')
```

5.3.5 setJCRPortalConnection

Module: Oracle WebCenter

Use with WLST: Online

5.3.5.1 Description

Edits an existing Oracle Portal connection. This command requires that you specify values for either the `dataSource` or `isPrimary` argument.

5.3.5.2 Syntax

```
setJCRPortalConnection(appName, name, [dataSource, extAppId, isPrimary, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing Oracle Portal connection.

Argument	Definition
dataSource	Optional. JNDI DataSource location used to connect to the portal. For example: `jdbc/MyPortalDS` The datasource must be on the server where the WebCenter application is deployed.
extAppId	Optional. External application used to authenticate WebCenter users against Oracle Portal. This value should match the name of an existing external application connection. See also listExtAppConnections. If `extAppId` is not set, no change is made to the authentication method or external application ID. If `extAppId` is set to an empty string, the authentication method used is `IDENTITY_PROPAGATION`. With this method, the WebCenter application and Oracle Portal use the same identity store to authenticate users.
isPrimary	Optional. Valid string values are `true` and `false`. `true` specifies that this connection is the primary connection used by the Documents service. When set to `false`, and the specified connection is the primary connection used by the Documents service, the primary connection is reset. If this parameter is not set, the primary connection used by the Documents service does not change. This argument has no default. In WebCenter Spaces, the primary connection must be an Oracle Content Server connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.5.3 Example

The following example edits Oracle Portal repository connection details.

```
wls:/weblogic/serverConfig> setJCRPortalConnection(appName='webcenter',
name='myPortalConnection', dataSource='/newPortalDS', extAppId='myExtApp',
isPrimary='false')
```

5.3.6 listJCRPortalConnections

Module: Oracle WebCenter

Use with WLST: Online

5.3.6.1 Description

Without any arguments, this command lists all of the Oracle Portal connections that are configured for a named WebCenter application.

5.3.6.2 Syntax

```
listJCRPortalConnections(appName, [verbose, name, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.

Argument	Definition
verbose	Optional. Displays content repository connection details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listJCRPortalConnections` lists all Oracle Portal connections that are configured for a WebCenter application, along with their details. When set to `false`, only connection names are listed. This argument defaults to `false`.
name	Optional. Name of an existing Oracle Portal connection. When specified you can view connection details for a specific Oracle Portal connection. If you supply a value for `name`, you must supply a value for `verbose`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.6.3 Example

The following example lists all of the Oracle Portal connections that are configured for a WebCenter application.

```
wls:/weblogic/serverConfig> listJCRPortalConnections(appName='webcenter',
verbose=true, name='myPortalConnection')
```

5.3.7 createJCRFileSystemConnection

Module: Oracle WebCenter

Use with WLST: Online

5.3.7.1 Description

Creates a connection to a file system repository.

> **Note:** File system connections *must not* be used in production or enterprise application deployments. This feature is provided for development purposes only.

5.3.7.2 Syntax

```
createJCRFileSystemConnection(appName, name, path, [isPrimary, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.
path	Full path to a folder whose contents you want to expose through this file system connection. For example, if you have a folder called `C:\ProjectDocuments` and you want to use that folder with the Documents service, you need to specify this folder as the path argument to this command.

Argument	Definition
isPrimary	Optional. Valid values are `true` and `false`. `true` specifies that this connection is the primary connection used by the Documents service. When set to `false`, and when the specified connection is the primary connection used by the Documents service, the primary connection is reset. If this parameter is not set, the primary connection used by the Documents service does not change. This argument has no default. In WebCenter Spaces, the primary connection must be an Oracle Content Server connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.7.3 Example

The following example creates a connection to a file system repository.

```
wls:/weblogic/serverConfig> createJCRFileSystemConnection(appName='webcenter',
name='FSAConnection', path='C:\ProjectDocuments')
```

5.3.8 setJCRFileSystemConnection

Module: Oracle WebCenter

Use with WLST: Online

5.3.8.1 Description

Edits an existing file system repository connection. This command requires that you specify values for either the `path` or `isPrimary` arguments.

> **Note:** File system connections *must not* be used in production or enterprise application deployments. This feature is provided for development purposes only.

5.3.8.2 Syntax

```
setJCRFileSystemConnection(appName, name, [path, [isPrimary, server,
applicationVersion])
```

Argument	Definition
appName	Application name in which you want to set Document service properties.
name	Name for the connection to be used by the Documents service.
path	Optional. Full path to a folder whose contents you want to expose through this file system connection. For example, if you have a folder called `C:\ProjectDocuments` and you want to use that folder with the Documents service, you need to specify this folder as the path argument to this command.

Argument	Definition
isPrimary	Optional. Valid values are `true` and `false`. When set to `true`, specifies that this connection is the primary connection used by the Documents service. When set to `false`, and when the specified connection is the primary connection used by the Documents service, the primary connection is reset. If this parameter is not set, the primary connection used by the Documents service does not change. This argument has no default. Note that in WebCenter Spaces, the primary connection must be an Oracle Content Server connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.8.3 Example

The following example edits connection details for a file system repository.

```
wls:/weblogic/serverConfig> setJCRFileSystemConnection(appName='webcenter',
name='FSAConnection', path='C:\ProjectDocuments')
```

5.3.9 listJCRFileSystemConnections

Module: Oracle WebCenter

Use with WLST: Online

5.3.9.1 Description

Without any arguments, this command lists all of the file system connections that are configured for a named WebCenter application.

> **Note:** File system connections *must not* be used in production or enterprise application deployments. This feature is provided for development purposes only.

5.3.9.2 Syntax

```
listJCRFileSystemConnections(appName, [verbose], [name, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
verbose	Optional. Displays content repository connection details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listJCRFileSystemConnections` lists all file system connections that are configured for a WebCenter application, along with their details. When set to `false`, only connection names are listed. This argument defaults to `false`.

Argument	Definition
name	Optional. Name of an existing file system connection. When specified you can view connection details for a specific file system connection. If you supply a value for `name`, you must supply a value for `verbose`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.9.3 Examples

The following example lists all of the file system connections that are configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig> listJCRFileSystemConnections(appName='webcenter')
```

The following example lists all of the file system connections that are configured, in verbose mode.

```
wls:/weblogic/serverConfig> listJCRFileSystemConnections(appName='webcenter',
verbose=true)
```

5.3.10 listDocumentsSpacesProperties

Module: Oracle WebCenter

Use with WLST: Online

5.3.10.1 Description

Lists properties for the back-end Oracle Content Server repository that is being used by WebCenter Spaces to store group space and personal space documents. This command is only valid for the WebCenter Spaces application.

5.3.10.2 Syntax

```
listDocumentsSpacesProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application in which you want to perform this operation—always `webcenter`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.10.3 Example

The following example lists properties for the back-end Oracle Content Server repository that is being used by a WebCenter Spaces application (named `webcenter`) to store group space and personal space documents.

```
wls:/weblogic/serverConfig> listDocumentsSpacesProperties(appName='webcenter')
```

5.3.11 setDocumentsSpacesProperties

Module: Oracle WebCenter

Use with WLST: Online

5.3.11.1 Description

Modifies properties for the back-end Oracle Content Server repository that is being used by WebCenter Spaces to store group space data. This command is only valid for the WebCenter Spaces application.

5.3.11.2 Syntax

```
setDocumentsSpacesProperties(appName, [spacesRoot, adminUserName,
applicationName, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application in which you want to perform this operation—always `webcenter`.
spacesRoot	Optional. Root folder under which group space content is stored. The value for this argument must use the format: `/foldername`. For example, `/WebCenter` or `/WebCenterSpaces`. The spacesRoot cannot be `/`, the root itself, and it must be unique across applications. If the folder specified does not exist it will be created for you. Note that if you provide a value for this argument, you must also provide values for the `adminUserName` and `applicationName` arguments.
adminUserName	Optional. User name of the content repository administrator. For example: `sysadmin`. Note that if you provide a value for this argument, you must also provide values for the `spacesRoot` and `applicationName` arguments. Administrative privileges are required for this connection so that operations can be performed on behalf of WebCenter users.
applicationName	Optional. Unique WebCenter Spaces application identifier. This name is used to separate data when multiple WebCenter Spaces applications share the same content repository, and must be unique across applications. The value for this argument must begin with an alphabetical character, followed by any combination of alphanumeric characters or the underscore character. The string must be less than or equal to 30 characters. Note that if you provide a value for this argument, you must also provide values for the `spacesRoot` and `adminUserName` arguments.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.11.3 Examples

The following example modifies connection properties for the back-end Oracle Content Server repository that is being used by WebCenter Spaces to store group space and personal space documents.

```
wls:/weblogic/serverConfig> setDocumentsSpacesProperties(appName='webcenter',
spacesRoot='/AccountingSpaces', adminUserName='admin',
applicationName='WCAccounting')
```

The following example modifies the administrator's user name for the back-end Oracle Content Server repository that is being used by WebCenter Spaces to store group space and personal space documents.

```
wls:/weblogic/serverConfig> setDocumentsSpacesProperties(appName='webcenter',
adminUserName='sysadmin')
```

5.3.12 deleteDocumentsSpacesProperties

Module: Oracle WebCenter

Use with WLST: Online

5.3.12.1 Description

Deletes properties for the back-end Oracle Content Server repository used by WebCenter Spaces, that is the `adminUserName`, `applicationName`, and `spacesRoot`. This command is only valid for the WebCenter Spaces application.

5.3.12.2 Syntax

```
deleteDocumentsSpacesProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application in which you want to perform this operation—always `webcenter`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.3.12.3 Example

The following example deletes connection properties (`adminUserName`, `applicationName`, `spacesRoot`) of the back-end Oracle Content Server repository that is being used by WebCenter Spaces.

```
wls:/weblogic/serverConfig> deleteDocumentsSpacesProperties(appName='webcenter')
```

5.4 Discussions and Announcements

Use the commands listed in Table 5–4 to manage discussions server connections for WebCenter applications.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 5–4 Discussion and Announcement WLST Commands

Use this command...	To...	Use with WLST...
createDiscussionForumConnection	Create a new discussions server connection for a WebCenter application.	Online
setDiscussionForumConnection	Edit an existing discussions server connection.	Online
setDiscussionForumConnectionProperty	Set an additional discussions server connection property.	Online
deleteDiscussionForumConnectionProperty	Delete a discussions server connection property.	Online
listDiscussionForumConnections	List all of the discussions server connections that are configured for an application.	Online
listDefaultDiscussionForumConnection	List the default discussions server connection for an application.	Online
setDefaultDiscussionForumConnection	Specify the default connection for the Discussions and Announcements services.	Online
setDiscussionForumServiceProperty	Specify defaults for the Discussions service.	Online
removeDiscussionForumServiceProperty	Remove defaults for the Discussions service.	Online
listDiscussionForumServiceProperties	List Discussions service properties.	Online
setAnnouncementServiceProperty	Specify defaults for the Announcements service.	Online
removeAnnouncementServiceProperty	Remove defaults for the Announcements service.	Online
listAnnouncementServiceProperties	List Announcements service properties.	Online

5.4.1 createDiscussionForumConnection

Module: Oracle WebCenter

Use with WLST: Online

5.4.1.1 Description

Creates a new discussions server connection for a named WebCenter application.

The Discussions service and the Announcements service both require a discussions server connection. Both services use the same discussions server connection.

While you can register multiple discussions server connections for a WebCenter application, only one connection is used for discussion and announcement services—the default (or active) connection.

5.4.1.2 Syntax

```
createDiscussionForumConnection(appName, name, url, adminUser, [timeout,
default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.
url	URL of the discussions server hosting discussion forums and announcements. For example: `http://myhost:8888/owc_discussions`.
adminUser	Name of the discussions server administrator. This account is used by the Discussions and Announcements services to perform administrative operations on behalf of WebCenter users. This account is mostly used for managing group space discussions and announcements in WebCenter Spaces. It is not necessary for this user to be a `super admin`. However, the user must have administrative privileges on the current application root category for WebCenter Spaces, that is, the category (on the discussions server) under which all group space discussions and announcements are stored.
timeout	Optional. Length of time (in seconds) the Discussions service waits for a response from the discussions server before issuing a connection timeout message. This argument defaults to `-1`. When set to `-1`, the service default (10 seconds) applies.
default	Optional. Indicates that this connection is the default connection for the Discussions and Announcements services. Valid options are `true` and `false`. When set to `true`, the Discussions service and the Announcements service both use this connection. When set to `false`, the connection is not used. The default is `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.1.3 Example

The following example creates a discussions server connection for a WebCenter application.

```
wls:/weblogic/serverConfig> createDiscussionForumConnection(appName='webcenter',
name='MyDiscussionServer', url='http://myhost.com:8888/owc_discussions',
adminUser='admin', default='false')
```

5.4.2 setDiscussionForumConnection

Module: Oracle WebCenter

Use with WLST: Online

5.4.2.1 Description

Edits an existing discussions server connection. Use this command to update connection attributes.

The connection is created using the createDiscussionForumConnection command.

5.4.2.2 Syntax

```
setDiscussionForumConnection(appName, name, [url, adminUser, timeout, default,
server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing discussions server connection.
url	Optional. URL to the discussions server.
adminUser	Optional. Name of the discussions server administrator. This account is used by the Discussions service to perform administrative operations on behalf of WebCenter users. This account is mostly used for managing group space discussions and announcements in WebCenter Spaces. It is not necessary for this user to be a `super admin`. However, the user must have administrative privileges on the current root category for WebCenter Spaces, that is, the category (on the discussions server) under which all WebCenter Spaces discussion forums are stored.
timeout	Optional. Length of time (in seconds) the Discussion and Announcement services wait for a response from the discussions server before issuing a connection timeout message. This argument defaults to `-1`. When set to `-1`, the service default (10 seconds) applies.
default	Optional. Indicates that this connection is the default connection for the Discussions and Announcements services. Required only if more than one connection is defined. Valid options are `true` and `false`. When set to `true`, the Discussion and Announcement services use this connection. When set to `false`, the connection is not used. The default is `false`. To specify that the Discussion and Announcements service use this connection, change the value from `false` to `true`. To disable this connection, use the removeDiscussionForumServiceProperty command: `removeDiscussionForumServiceProperty('appName='webcenter', 'property='selected.connection')` Note: While you can register multiple discussions server connections for a WebCenter application, only one connection is used for discussion and announcement services— the default (or active) connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.2.3 Example

The following example updates attributes for a discussions server connection named `MyDiscussionsServer`.

```
wls:/weblogic/serverConfig> setDiscussionForumConnection(appName='webcenter',
name='MyDiscussionServer', url='http://myhost.com:7786/owc_discussions',
adminUser='admin', default=true)
```

5.4.3 setDiscussionForumConnectionProperty

Module: Oracle WebCenter

Use with WLST: Online

5.4.3.1 Description

Set a discussions server connection property. Use this command if additional parameters are required to connect to your discussions server. This is an extensible way to add any connection property using a key and a value. (You are not limited to connection properties specified by createDiscussionForumConnection and setDiscussionForumConnection.)

Note: Do not use the setDiscussionForumConnectionProperty to set connection properties available through createDiscussionForumConnection or setDiscussionForumConnection. Attempting to do so, has no effect.

All known, additional connection properties are listed in Table 5–5, " Additional Discussion Connection Properties".

Table 5–5 Additional Discussion Connection Properties

Additional Connection Property	Description
forum.connection.secure	Indicates whether secure communication is required between the WebCenter application and the discussions server. Valid values are `true` and `false`. When set to `true` (secured mode), all WebService calls from the WebCenter application are sent with a user name token and client certificate.
keystore.location	Certificate file path in your local directory.
keystore.password	Keystore password.
keystore.type	Keystore type associated with the certificate. Valid values are: `jks` (Java Key Store) and `pks`.
encryption.key.alias	Key alias to be used for encryption.
encryption.key.password	Password for accessing the encryption key.
group.mapping	(WebCenter Spaces only) Determines whether a subcategory or a single forum is created on the discussions server for new group spaces. When set to `forum` (default), a single forum is created under the application's root category per group space. When set to `category`, a subcategory is created under the application root category per group space. When a subcategory that supports multiple forums is more suitable, use setDiscussionForumConnectionProperty to set the `group.mapping` property to `category`. If a group space template has been configured with a forum-based taxonomy, then the template takes precedence over this mapping. If a group space template does not define the mapping (the Blank template, for example) then the `group.mapping` property is used. If there is no value in the template or the connection, the default setting is used (`forum`).

5.4.3.2 Syntax

```
setDiscussionForumConnectionProperty(appName, name, key, value, [secure, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing discussions server connection.
key	Name of the connection property.
value	Value for the property. Allows any property to be modified on the connection with a key and value.
secure	Optional. Indicates whether the property value must be stored securely using encryption. Valid options are `true` and `false`. When `true`, the value is encrypted. The default option is `false`. If this is set to `true`, the discussions server connection should be secured with WS-Security. Once marked secured, some mandatory properties have to be set otherwise this is a corrupt connection setting.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.3.3 Example

The following example adds a custom discussions server connection property called `myProperty1` with a value `propertyValue1`.

```
wls:/weblogic/serverConfig> setDiscussionForumConnectionProperty
(appName='webcenter', name='MyDiscussionServer', key='myProperty1',
value='propertyValue1')
```

5.4.4 deleteDiscussionForumConnectionProperty

Module: Oracle WebCenter

Use with WLST: Online

5.4.4.1 Description

Deletes a discussions server connection property. Take care when deleting connection properties because the connection may not work as expected if the configuration becomes invalid as a result.

This command can only delete *additional* connection properties added using the `setDiscussionForumConnectionProperty` command.

5.4.4.2 Syntax

```
deleteDiscussionForumConnectionProperty(appName, name, key, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing discussions server connection.
key	Name of the connection property you want to delete.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.4.3 Example

The following example deletes a discussions server connection property named `myProperty1`.

```
wls:/weblogic/serverConfig> deleteDiscussionForumConnectionProperty
(appName='webcenter', name='MyDiscussionServer', key='myProperty1')
```

5.4.5 listDiscussionForumConnections

Module: Oracle WebCenter

Use with WLST: Online

5.4.5.1 Description

Lists all of the discussions server connections that are configured for a named WebCenter application.

5.4.5.2 Syntax

```
listDiscussionForumConnections(appName, [name, verbose, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Optional. Name of an existing discussions server connection. Use this argument to view connection details for a specific discussions server connection.
verbose	Optional. Valid options are `true` and `false`. When set to `true`, `listDiscussionForumConnections` lists all of the discussions server connections that are configured for a WebCenter application, along with their details. When set to `false`, only connection names are listed. This argument defaults to `false`. If you use the `name` argument when `verbose` is set to `true`, the `verbose` argument is ignored.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.

Argument	Definition
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.5.3 Examples

The following example lists the names of all of the discussions server connections that are currently configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig>listDiscussionForumConnections(appName='webcenter')
```

The following example lists connection names and details for all of the discussions server connections currently configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig>listDiscussionForumConnections(appName='webcenter',
verbose=true)
```

The following example lists connection details for a discussions server connection named `myDiscussionsServer`.

```
wls:/weblogic/serverConfig> listDiscussionForumConnections(appName='webcenter',
name='myDiscussionsServer')
```

5.4.6 listDefaultDiscussionForumConnection

Module: Oracle WebCenter

Use with WLST: Online

5.4.6.1 Description

Names the discussions server connection that the Discussions service and the Announcements service are using, in a named WebCenter application. While you can register multiple discussions server connections for a WebCenter application, the Discussions/Announcements service only uses one connection—known as the default (or active) connection.

5.4.6.2 Syntax

```
listDefaultDiscussionForumConnection(appName, [verbose, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
verbose	Optional. Valid options are `true` and `false`. When set to `true`, the name and details of the discussions server connections are listed. When set to `false`, only the connection name displays. This argument defaults to `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.6.3 Examples

The following example names the discussions server connection that the Discussions/Announcements service are using, in an application named `webcenter`.

```
wls:/weblogic/serverConfig>
listDefaultDiscussionForumConnection(appName='webcenter')
```

The following example lists the name and details of the discussions server connection that the Discussions/Announcements service are using.

```
wls:/weblogic/serverConfig>
listDefaultDiscussionForumConnection(appName='webcenter', verbose=true)
```

5.4.7 setDefaultDiscussionForumConnection

Module: Oracle WebCenter

Use with WLST: Online

5.4.7.1 Description

Specifies the *default* discussions server connection for the Discussions service and the Announcements service, in a named WebCenter application.

While you can register multiple discussions server connections with a WebCenter application, the Discussions/Announcements services only uses one connection—this is known as the default (or active) connection.

5.4.7.2 Syntax

```
setDefaultDiscussionForumConnection(appName, name, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing discussions server connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.7.3 Example

The following example makes a connection named `myDiscussionServer` the default (or active) connection for the Discussions and Announcement services.

```
wls:/weblogic/serverConfig> setDefaultDiscussionForumConnection
(appName='webcenter', name='myDiscussionServer')
```

5.4.8 setDiscussionForumServiceProperty

Module: Oracle WebCenter

Use with WLST: Online

5.4.8.1 Description

Specifies default values for the Discussions service.

Configurable properties for the Discussions service are listed in Table 5–6, " Discussion Service Configuration Properties".

Table 5–6 Discussion Service Configuration Properties

Configuration Property	Description
topics.fetch.size	Maximum number of topics fetched by the Discussions service and displayed in the topics view.
forums.fetch.size	Maximum number of forums fetched by the Discussions service and displayed in the forums view.
recentTopics.fetch.size	Maximum number of topics fetched by the Discussions service and displayed in the recent topics view.
watchedTopics.fetch.size	Maximum number of topics fetched by the Discussions service and displayed in the watched topics view.
watchedForums.fetch.size	Maximum number of forums fetched by the Discussions service and displayed in the watched forums view.
application.root.category.id	Application root category ID on the discussions server under which all discussion forums are stored. For example, if set to 3, all forums are stored inside category 3.

5.4.8.2 Syntax

```
setDiscussionForumServiceProperty(appName, property, value, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
property	Name of the configuration property.
value	Value for the property.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.8.3 Example

The following example changes the default number of topics displayed in topics view.

```
wls:/weblogic/serverConfig>setDiscussionForumServiceProperty
(appName='webcenter', property='topics.fetch.size', value='30')
```

5.4.9 removeDiscussionForumServiceProperty

Module: Oracle WebCenter

Use with WLST: Online

5.4.9.1 Description

Removes default values for the Discussions service. Note that removing a default property might cause unexpected behavior.

> **Note:** Use this command syntax to disable the connection currently used for discussion and announcement services:
>
> `removeDiscussionForumServiceProperty('appName='webcenter', 'property='selected.connection')`
>
> This command forces the `default` connection argument to `false`. See also, setDiscussionForumConnection.

5.4.9.2 Syntax

```
removeDiscussionForumServiceProperty(appName, property, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
property	Name of the configuration property.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.9.3 Example

The following example clears the current `topics.fetch.size` property for the Discussions service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig> removeDiscussionForumServiceProperty
(appName='webcenter', property='topics.fetch.size')
```

5.4.10 listDiscussionForumServiceProperties

Module: Oracle WebCenter

Use with WLST: Online

5.4.10.1 Description

Lists all configurable properties for the Discussions service.

5.4.10.2 Syntax

```
listDiscussionForumServiceProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.10.3 Example

The following example lists configuration properties for the Discussions service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig>
listDiscussionForumServiceProperties(appName='webcenter')
```

5.4.11 setAnnouncementServiceProperty

Module: Oracle WebCenter

Use with WLST: Online

5.4.11.1 Description

Specifies default values for the Announcements service.

Configurable properties for the Announcements service are listed in Table 5–7, " Announcements Service Configuration Properties".

Table 5–7 Announcements Service Configuration Properties

Configuration Property	Description
miniview.page_size	Maximum number of announcements displayed in the Announcements mini view.
mainview.page_size	Maximum number of announcements displayed in the Announcements main view.
linksview.page_size	Maximum number of announcements displayed in the Announcements links view.
announcements.expiration.days	Number of days that announcements display and remain editable.

5.4.11.2 Syntax

```
setAnnouncementServiceProperty(appName, property, value, [server,
applicationVersion})
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
property	Name of the configuration property.
value	Property value.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.

Argument	Definition
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.11.3 Example

The following example changes the default number of days that announcements display, in an application named `webcenter`.

```
wls:/weblogic/serverConfig> setAnnouncementServiceProperty(appName='webcenter',
property='announcements.expiration.days', value='21')
```

5.4.12 removeAnnouncementServiceProperty

Module: Oracle WebCenter

Use with WLST: Online

5.4.12.1 Description

Removes default values for the Announcements service. Note that removing a default property might cause unexpected behavior.

5.4.12.2 Syntax

```
removeAnnouncementServiceProperty(appName, property, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
property	Name of the configuration property.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.12.3 Example

The following example clears the `announcements.expiration.days` property for the Announcements service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig> removeAnnouncementServiceProperty
(appName='webcenter', property='announcements.expiration.days')
```

5.4.13 listAnnouncementServiceProperties

Module: Oracle WebCenter

Use with WLST: Online

5.4.13.1 Description

Lists all configurable properties for the Announcements service.

5.4.13.2 Syntax

```
listAnnouncementServiceProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.4.13.3 Example

The following example lists configuration properties for the Announcements service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig> listAnnouncementServiceProperties(appName='webcenter')
```

5.5 External Applications

Use the commands listed in Table 5–8 to manage external application connections for WebCenter applications.

Configuration changes made using these WebCenter WLST commands are immediately available in the WebCenter application.

Table 5–8 External Application WLST Commands

Use this command...	To...	Use with WLST...
createExtAppConnection	Create an external application connection, for a named WebCenter application.	Online
setExtAppConnection	Edit an existing external application connection.	Online
listExtAppConnections	List individual or all external applications that are configured for a specific WebCenter application.	Online
addExtAppField	Add another login field for a specific external application connection.	Online
setExtAppField	Edit the value and display-to-user setting for a specific external application login field.	Online
removeExtAppField	Remove an external application login field.	Online
addExtAppCredential	Specify shared or public credentials for an external application.	Online
setExtAppCredential	Edit shared or public credentials for an external application.	Online
removeExtAppCredential	Remove shared or public credentials currently configured for an external application.	Online

5.5.1 createExtAppConnection

Module: Oracle WebCenter

Use with WLST: Online

5.5.1.1 Description

Creates an external application connection, for a named WebCenter application.

5.5.1.2 Syntax

```
createExtAppConnection(appName, name, [displayName, url, authMethod,
userFieldName, pwdFieldName, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.
displayName	Optional. External application display name. A user friendly name for the application that WebCenter users will recognize. The display name must be unique across all external applications within the WebCenter application.
url	Optional. External application login URL. To determine an application's URL, navigate to the application's login page and note down the URL for that page. For example: `http://login.yahoo.com/config/login`
authMethod	Optional. Authentication mechanism used by the external application. Valid options are `GET`, `POST`, and `BASIC`. This argument defaults to `POST`.
userFieldName	Optional. Name that identifies the *user name* or *user ID* field on the external application's login form. To find this name, look at the HTML source for the login page. This argument does not specify user credentials. Mandatory if `authMethod` is `GET` or `POST`, but can be left blank if `BASIC` authentication method is selected.
pwdFieldName	Optional. Name that identifies the *password* field on the external application's login form. To find this name, look at the HTML source for the login page. This argument does not specify user credentials. Mandatory if `authMethod` is `GET` or `POST`, but can be left blank if `BASIC` authentication method is selected.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.5.1.3 Example

The following example creates a connection for an external application named `My Yahoo!`, in a WebCenter application.

```
wls:/weblogic/serverConfig> createExtAppConnection(appName='webcenter',
name='yahoo', displayName='My Yahoo!', url='http://login.yahoo.com/config/login',
authMethod='POST', userFieldName='login', pwdFieldName='passwd')
```

5.5.2 setExtAppConnection

Module: Oracle WebCenter

Use with WLST: Online

5.5.2.1 Description

Edits an existing external application connection.

5.5.2.2 Syntax

```
setExtAppConnection(appName, name, [displayName], [url], [authMethod],
[userFieldName], [pwdFieldName], [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing external application connection.
displayName	Optional. External application display name. A user-friendly name for the application that WebCenter users will recognize.
url	Optional. External application login URL. To determine an application's URL, navigate to the application's login page and note down the URL for that page.
authMethod	Optional. Authentication mechanism used by the external application. Valid options are `GET`, `POST`, and `BASIC`. This argument defaults to `POST`.
userFieldName	Optional. Name that identifies the *user name* or *user ID* field on the external application's login form. To find this name, look at the HTML source for the login page. This argument does not specify user credentials. Mandatory if `authMethod` is `GET` or `POST`, but can be left blank if `BASIC` authentication method is selected.
pwdFieldName	Optional. Name that identifies the *password* field on the external application's login form. To find this name, look at the HTML source for the login page. This argument does not specify user credentials. Mandatory if `authMethod` is `GET` or `POST`, but can be left blank if `BASIC` authentication method is selected.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.5.2.3 Example

The following example updates the display name attribute for an external application named `yahoo`.

```
wls:/weblogic/serverConfig> setExtAppConnection(appName='webcenter',
name='yahoo', displayName='My Favorite Yahoo!')
```

5.5.3 listExtAppConnections

Module: Oracle WebCenter

Use with WLST: Online

5.5.3.1 Description

When used with only the `appName` argument, this command lists the names of all the external applications currently configured for a specific WebCenter application.

5.5.3.2 Syntax

```
listExtAppConnections(appName, [verbose, name, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application for which you want to perform this operation.
verbose	Optional. Displays external application details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listExtAppConnections` lists all of the external applications that are configured for a WebCenter application, along with their details. When set to `false`, `listExtAppConnections` lists only the names of the external applications. This argument defaults to `false`. If you set this argument to `false`, do not specify the `name` argument.
name	Optional. Name of an existing external application connection. You can use this argument to view details about a specific connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.5.3.3 Examples

The following example lists the names of all the external applications currently used by a WebCenter application named `webcenter`.

```
wls:/weblogic/serverConfig> listExtAppConnections(appName='webcenter')
app1
app2
app3
```

The following example lists details for the external applications `app1`, `app2`, and `app3`.

```
wls:/weblogic/serverConfig> listExtAppConnections(appName='webcenter',
verbose=true)
----
app1
----
Name: app1
Display Name: Application1
Login URL: http://app1
Authentication Method: POST
User Field Name: login
Password Field Name: passwd
Shared Credential: Disabled
Public Credential: Disabled
----
app2
----
Name: app2
```

```
Display Name: Application2
Login URL: http://app2
Authentication Method: POST
User Field Name: login
Password Field Name: passwd
Additional Fields: {Account1:true, Accout2:DefVal:false}
Shared Credential: Disabled
Public Credential: Enabled
----
app3
----
Name: app3
Display Name: Application3
Authentication Method: POST
Shared Credential: Enabled
Public Credential: Enabled
```

The following example lists details for external application `app1` only.

```
wls:/weblogic/serverConfig> listExtAppConnections(appName='webcenter',
verbose=true, name='app1')
----
app1
----
Name: app1
Display Name: Application1
Login URL: http://app1
Authentication Method: POST
User Field Name: login
Password Field Name: passwd
Shared Credential: Disabled
Public Credential: Disabled
```

5.5.4 addExtAppField

Module: Oracle WebCenter

Use with WLST: Online

5.5.4.1 Description

Adds another login field for a specific external application connection. For example, in addition to user name and password, an external application may require other login criteria such as Host and MailAddress.

Optionally, additional login fields can appear on the external application's login for a user to specify.

If you add another login field *and* the external application uses shared or public credentials, you can use the WLST commands and `setExtAppCredential` to update the shared/public credentials. See Section 5.5.7, "addExtAppCredential" and Section 5.5.8, "setExtAppCredential".

5.5.4.2 Syntax

```
addExtAppField(appName, name, fieldName, [fieldValue], [displayToUser], [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing external application connection.
fieldName	Login field name. The name that identifies the field on the HTML login form. This field is not applicable if the application uses BASIC authentication.
fieldValue	Optional. Login field value. Enter a default value for the login field or leave blank for a user to specify. This argument is blank by default.
displayToUser	Optional. Specifies whether the login field displays on the external application's login screen. Valid options are `true` and `false`. This argument defaults to `false`. Note that if you set this argument to `false`, you must specify the `fieldValue`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.5.4.3 Example

This example creates an additional field named `Account` with the default value `username.default.example` in an external application called `ABC`. This field will be displayed in `ABC`'s login screen.

```
wls:/weblogic/serverConfig> addExtAppField(appName='webcenter', name='ABC',
fieldName='Account', fieldValue='username.default.example',
displayToUser=true)
```

5.5.5 setExtAppField

Module: Oracle WebCenter

Use with WLST: Online

5.5.5.1 Description

Modifies the field value and display-to-user setting for one or more login fields currently configured for an external application. Either `fieldValue` or `displayToUser` must be specified along with the external application name and login field name. The `fieldValue` and `displayToUser` arguments are optional.

Using this command has implications on any shared or public credentials that you might have created for this external application. If you modify `displayToUser` to `true`, you may also need to update existing shared user or public user credentials. See also Section 5.5.8, "setExtAppCredential".

5.5.5.2 Syntax

```
setExtAppField(appName, name, fieldName, [fieldValue], [displayToUser], [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing external application connection.
fieldName	Name of an existing login field.
fieldValue	Optional. New or changed login field value. Enter a default value for the login field or leave blank for a user to specify. This argument is blank by default.
displayToUser	Optional. Specifies whether the login field displays on the external application's login screen. Valid options are `true` and `false`. If set to `false`, `fieldValue` must be specified.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.5.5.3 Example

The following example specifies a default value for a login field named `Account` and displays the field on the external application's credential provisioning screen.

```
wls:/weblogic/serverConfig> setExtAppField(appName='webcenter', name='ABC',
fieldName='Account', fieldValue='admin', displayToUser=true)
```

5.5.6 removeExtAppField

Module: Oracle WebCenter

Use with WLST: Online

5.5.6.1 Description

Removes a login field from an external application connection.

This command has implications on any shared or public credentials that you may have created for this external application, that is, you may need to remove the login field from shared user or public user credentials.

You can use the `setExtAppCredential` command to remove a login field, if required. For example, external application `myApp` has an additional field called `Account` and public credentials were previously specified using:

```
addExtAppCredential(appName='webcenter', name='myApp', type='PUBLIC',
username='admin', password='mypublic.password', field='Account:admin@myhost.com')
```

If you remove the `Account` field, you can modify the credentials by running:

```
setExtAppCredential(appName='webcenter', name='myApp', type='PUBLIC',
username='admin', password='mypublic.password')
```

For details on using `setExtAppCredential`, see Section 5.5.8, "setExtAppCredential"

5.5.6.2 Syntax

```
removeExtAppField(appName, name, fieldName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name.
fieldName	Login field that you want to remove.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.5.6.3 Example

The following example removes the additional login field named `Account` from an external application named `ABC`.

```
wls:/weblogic/serverConfig> removeExtAppField(appName='webcenter, name='ABC',
fieldName='Account')
```

5.5.7 addExtAppCredential

Module: Oracle WebCenter

Use with WLST: Online

5.5.7.1 Description

Configures shared user or public user credentials for a specific external application.

When shared credentials are specified, every user accessing the WebCenter application is authenticated using the user name and password defined here. WebCenter users are not presented with a login form.

Public users accessing this application through WebCenter are logged in using the user name and password defined here.

If credentials already exists, a warning indicates that the `setExtAppCredential` command should be used instead.

5.5.7.2 Syntax

```
addExtAppCredential(appName, name, type, username, password, [field, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing external application connection.
type	Credential type. Valid values are `SHARED` and `PUBLIC`.
username	Name of the shared or public user.
password	Password for the shared or public user.

Argument	Definition
field	Optional. Additional login field value. Use the format `FieldName:FieldValue`, where `FieldName` names an additional login field configured with `displayToUser=true`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.5.7.3 Example

The following example specifies public credentials for an external application named `ABC`. The public user name is `mypublic.username`, the password is `mypublic.password`, and there is one additional field named `Account`.

```
wls:/weblogic/serverConfig> addExtAppCredential(appName='webcenter', name='ABC',
type='PUBLIC', username='mypublic.username', password='mypublic.password',
field='Account:username.example')
```

5.5.8 setExtAppCredential

Module: Oracle WebCenter

Use with WLST: Online

5.5.8.1 Description

Modifies shared user or public user credentials currently configured for an external application. If the credential has already not been specified, then a warning indicates that `addExtAppCredential` needs to be used instead. See Section 5.5.7, "addExtAppCredential".

The arguments `username` and `password` are optional because `setExtAppCredential` only manipulates existing credentials.

You can use `setExtAppCredential` command to update passwords in systems that require changing passwords every few days.

5.5.8.2 Syntax

```
setExtAppCredential(appName, name, type, [username], [password], [field],
[server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing external application connection.
type	Credential type. Valid values are `SHARED` and `PUBLIC`.
username	Optional. User name of the shared or public user.
password	Optional. Password for the shared or public user.
field	Optional. Additional login field value. Use the format `FieldName:FieldValue`, where `FieldName` names an additional login field configured with `displayToUser=true`.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.5.8.3 Example

The following example changes the public user's login credentials for an external application named ABC.

```
wls:/weblogic/serverConfig> setExtAppCredential(appName='webcenter',name='ABC',
type='PUBLIC', username='username.example', password='password.example',
field='Account:username.example')
```

5.5.9 removeExtAppCredential

Module: Oracle WebCenter

Use with WLST: Online

5.5.9.1 Description

Removes shared user or public user credentials currently configured for an external application.

If credentials do not exist, an error displays.

5.5.9.2 Syntax

```
removeExtAppCredential(appName, name, type, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing external application connection.
type	Credential type. Valid values are SHARED and PUBLIC.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.5.9.3 Example

The following example removes shared credentials specified for an external application named ABC.

```
wls:/weblogic/serverConfig> removeExtAppCredential(appName='webcenter',
name='ABC', type='SHARED')
```

5.6 Instant Messaging and Presence

Use the commands listed in Table 5–9, to manage instant messaging and presence server connections.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 5–9 Instant Messaging and Presence WLST Commands

Use this command...	To...	Use with WLST...
createIMPConnection	Create a new instant messaging and presence server connection for a WebCenter application.	Online
setIMPConnection	Edit an existing instant messaging and presence server connection.	Online
setIMPConnectionProperty	Modify instant messaging and presence server connection properties.	Online
deleteIMPConnectionProperty	Delete an instant messaging and presence server connection property.	Online
listIMPAdapters	List which presence servers the WebCenter application supports.	Online
listIMPConnections	List all of the instant messaging and presence server connections that are configured for an application.	Online
listDefaultIMPConnection	List the default instant messaging and presence server connection that is configured for an application.	Online
setDefaultIMPConnection	Set a specified connection as the default instant messaging and presence server connection.	Online
setIMPServiceProperty	Specify defaults for the Instant Messaging and Presence service.	Online
removeIMPServiceProperty	Remove defaults for the Instant Messaging and Presence service.	Online
listIMPServiceProperties	List Instant Messaging and Presence service properties.	Online

5.6.1 createIMPConnection

Module: Oracle WebCenter

Use with WLST: Online

5.6.1.1 Description

Creates an instant messaging and presence server connection for a named WebCenter application.

Use the listIMPAdapters command to find out which types of instant messaging and presence servers are supported. Out-of-the-box, WebCenter applications support Oracle WebLogic Communication Server (OWLCS) and Microsoft Live Communication Server (LCS).

While you can register multiple presence server connections for a WebCenter application, only one connection is used for instant messaging and presence services—the default (or active) connection.

5.6.1.2 Syntax

```
createIMPConnection(appName, name, adapter, url, domain, [appId, poolName,
policyURI, timeout, default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.
url	URL of the sever hosting instant messaging and presence services. For example: `http://myowlcshost.com:8888`
adapter	Adapter name. Specify the adapter that matches your instant messaging and presence server. Valid values are `LCS` and `OWLCS`. Choose `LCS` for Microsoft Live Communication Server. Choose `OWLCS` for Oracle WebLogic Communication Server.
domain	User domain associated with this connection. The domain specified is used to construct each user's SIP ID. For example, if the domain is `oracle.com` and presence is requested for user with name `john` then the SIP address resolved will be `sip:john@oracle.com`. If the user SIP address needs to be resolved from the OID/LDAP server, then specify the user profile attribute that will provide the SIP address here as `profile:<attribute>` where *profile* is a keyword and *attribute* is the user profile attribute name where the SIP address is stored. For example, `profile:primarySipAddress`. SIP is short for Session Initiation Protocol - an Internet protocol for live communication between people.
appId	Optional. External application associated with the presence server connection. If specified, external application credential information is used to authenticate users against the LCS or OWLCS server. This argument is mandatory for LCS server connections. The external application you configure for the IMP service must use `authMethod=POST`, and specify an additional field with `fieldName='Account'` and `displaytoUser=true`. See also addExtAppField and setExtAppField.
poolName	Optional. (LCS connections only.) Pool name that is required to create an LCS connection. Refer to *Microsoft Live Communication Server* documentation for details on the pool name. This argument is mandatory for LCS server connections.
policyURI	Optional. (OWLCS connections only.) URI to the security policy that is required for authentication on the Oracle WebLogic Communication Server (OWLCS) server.
timeout	Optional. Length of time (in seconds) that the Instant Messaging and Presence service waits for a response from the presence server before issuing a connection timeout message. This argument defaults to `-1`. When set to `-1`, the service default (10 seconds) applies.

Argument	Definition
default	Optional. Indicates whether this connection is the default connection for the Instant Messaging and Presence service. Valid values are `true` and `false`. The default for this argument is `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.1.3 Examples

The following example creates an instant messaging and presence server connection to a Oracle WebLogic Communication Server named `myOWLCSPresenceServer`.

```
wls:/weblogic/serverConfig>createIMPConnection(appName='webcenter',
name='myOWLCSPresenceServer', adapter='OWLCS',
url='http://myowlcshost.com:8888', domain='oracle.com')
```

The following example also creates an instant messaging and presence server connection to a Oracle WebLogic Communication Server named `myOWLCSPresenceServer`.

```
wls:/weblogic/serverConfig>createIMPConnection(appName='webcenter',
name='myOWLCSPresenceServer', adapter='OWCLS', url='http://myowlcshost.com:8888',
domain='oracle.com', policyURI='oracle/wss11_saml_token_with_message_
protection_client_policy', timeout=60, default=false)
```

The following example creates an instant messaging and presence server connection to a Microsoft Live Communication Server named `myLCSPresenceServer`.

```
wls:/weblogic/serverConfig> createIMPConnection(appName='webcenter',
name='myLCSPresenceServer', adapter='LCS', url='http://mylcshost.com/owc/lcs',
domain='oracle.com', appId='LCSExtApp', poolName='pool1.myhost.com', timeout=60,
default=true)
```

5.6.2 setIMPConnection

Module: Oracle WebCenter

Use with WLST: Online

5.6.2.1 Description

Edits an existing instant messaging and presence server connection. Use this command to update connection attributes.

The connection is created using the createIMPConnection command.

5.6.2.2 Syntax

```
setIMPConnection(appName, name, [adapter, url, domain, appId, poolName,
policyURI, timeout, default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.

Argument	Definition
name	Name of an existing presence server connection.
adapter	Optional. Adapter name. Specify the adapter that matches your instant messaging and presence server. Valid values are `LCS` and `OWLCS`. Choose `LCS` for Microsoft Live Communication Server. Choose `OWLCS` for Oracle WebLogic Communication Server.
url	Optional. URL of the server hosting instant messaging and presence services.
domain	Optional. User domain associated with this connection. The domain specified is used to construct each user's SIP ID. For example, if the domain is `oracle.com` and presence is requested for user with name `john` then the SIP address resolved will be `sip:john@oracle.com`. If the user SIP address needs to be resolved from the OID/LDAP server, then specify the user profile attribute that will provide the SIP address here as `profile:<attribute>` where *profile* is a keyword and *attribute* is the user profile attribute name where the SIP address is stored. For example, `profile:primarySipAddress`. Alternatively, dynamically resolve the user SIP address on your own using a custom class. Provide a custom class that implements `oracle.webcenter.collab.rtc.IMPAddressResolver` and then specify the custom class here as `custom:com.company.custom.AddressResolver` (assuming your class is `com.company.custom.AddressResolver`). SIP is short for Session Initiation Protocol - an Internet protocol for live communication between people.
appId	Optional. External application associated with the presence server connection. If specified, external application credential information is used to authenticate users against the LCS or OWLCS server. This argument is mandatory for LCS server connections. The external application you configure for the IMP service must use `authMethod=POST`, and specify an additional field with `fieldName='Account'` and `displaytoUser=true`. See also addExtAppField and setExtAppField.
poolName	Optional. (LCS connections only.) Pool name that is required to create an LCS connection. Mandatory for LCS connections. Refer to *Microsoft Live Communication Server* documentation for details on the pool name.
policyURI	Optional. (OWLCS connections only.) URI to the security policy that is required for authentication on the Oracle WebLogic Communication Server (OWLCS) server.
timeout	Optional. Length of time (in seconds) that the Instant Messaging and Presence service waits for a response from the presence server before issuing a connection timeout message. This argument defaults to `-1`. When set to `-1`, the service default (10 seconds) applies.

Argument	Definition
default	Optional. Indicates whether this connection is the default connection for the Instant Messaging and Presence service. Valid values are `true` and `false`. The default for this argument is `false`. To specify that the Instant Messaging and Presence service uses this connection, change the value from `false` to `true`. To disable this connection, use the removeIMPServiceProperty command: `removeIMPServiceProperty('appName='webcenter', 'property='selected.connection')` While you can register multiple presence server connections for a WebCenter application, only one connection is used for instant messaging and presence services— the default (or active) connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.2.3 Examples

The following example updates attributes for an existing instant messaging and presence server connection.

```
wls:/weblogic/serverConfig>setIMPConnection(appName='webcenter',
name='myOWLCSPresenceServer', adapter='OWCLS', url='http://myowlcshost.com:8888',
domain='oracle.com')
```

The following example sets attributes on an existing instant messaging and presence server connection.

```
wls:/weblogic/serverConfig>setIMPConnection(appName='webcenter',
name='myOWLCSPresenceServer', adapter='OWCLS', url='http://myowlcshost.com:8888',
domain='oracle.com', policyURI='oracle/wss11_saml_token_with_message_protection_
client_policy', timeout=60, default=false)
```

The following example sets attributes on an existing instant messaging and presence server connection.

```
wls:/weblogic/serverConfig>setIMPConnection(appName='webcenter',
name='myLCSPresenceServer', adapter='LCS', url='http://mylcshost.com/owc/lcs',
domain='oracle.com', appId='LCSExtApp', poolName='pool1.myhost.com', timeout=60,
default=false)
```

5.6.3 setIMPConnectionProperty

Module: Oracle WebCenter

Use with WLST: Online

5.6.3.1 Description

Sets an instant messaging and presence server connection property. Use this command if additional parameters are required to connect to your presence server. This is an extensible way to add any connection property using a key and a value. (You are not limited to connection properties specified by createIMPConnection and setIMPConnection.)

All known, additional connection properties are listed in Table 5–10, " Additional IMP Connection Properties".

Table 5–10 Additional IMP Connection Properties

Additional Connection Property	Description
presence.url	(OWLCS only) URL to the OWLCS Presence service. Required if the OWLCS Presence service is deployed on a separate node. When no value is specified, the connection `url` property is used.
contacts.url	(OWLCS only) URL to the OWLCS Contact Management service. Required if the OWLCS Contact Management service is deployed on a separate node. When no value is specified, the connection `url` property is used.
call.url	(OWLCS only) URL to the OWLCS Third Party Call service. Required if the OWLCS Third Party Call service is deployed on a separate node. When no value is specified, the connection's `url` property is used.
call.method	(OWLCS only) Third party call method. Valid values are: `sip` and `pstn`. When set to `sip`, third party calls are forwarded to Oracle Communicators. When set to pstn, calls are forwarded to PSTN telephones (`contact.number.attribute` provides the phone number).
`call.domain`	(OWLCS only) Domain name of the PSTN gateway. Required when the `call.method` is `pstn`. If no domain name is supplied, the connection's `domain` value is used.
contact.number.attribute	(OWLCS only) User profile attribute used to store users' phone numbers. The default attribute is `BUSINESS_PHONE`. Required when the call.method is `pstn`.
primary.domain	(OWLCS and LCS) User domain. If WebCenter usernames are qualified with a domain, specify that domain here. For example, when usernames are `xyz@example.com`, the `primary.domain` is `example.com`.

Do not use the setIMPConnectionProperty to set connection properties available through createIMPConnection or setIMPConnection. Attempting to do so, has no effect.

5.6.3.2 Syntax

```
setIMPConnectionProperty(appName, name, key, value, [secure, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing presence server connection.
key	Name of the connection property.
value	Value for the property. Allows any property to be modified on the connection with a key and value.

Argument	Definition
secure	Optional. Indicates whether the property value must be stored securely using encryption. Valid options are `true` and `false`. When `true`, the value is encrypted. The default option is `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.3.3 Example

The following example adds a custom instant messaging and presence server connection property called `admin.user` with a default value `admin`.

```
wls:/weblogic/serverConfig> setIMPConnectionProperty(appName='webcenter',
name='MyLCSPresenceServer', key='admin.user', value='admin')
```

5.6.4 deleteIMPConnectionProperty

Module: Oracle WebCenter

Use with WLST: Online

5.6.4.1 Description

Deletes an instant messaging and presence server connection property. Use caution when deleting connection properties because the connection might not work as expected if the configuration becomes invalid as a result.

This command can only delete *additional* connection properties added using the `setIMPConnectionProperty` command.

5.6.4.2 Syntax

```
deleteIMPConnectionProperty(appName, name, key, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing presence server connection.
key	Name of the connection property you want to delete.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.4.3 Example

The following example deletes an instant messaging and presence server connection property named `admin.user`.

```
wls:/weblogic/serverConfig> deleteIMPConnectionProperty(appName='webcenter',
```

```
name='MyLCSPresenceServer', key='admin.user')
```

5.6.5 listIMPAdapters

Module: Oracle WebCenter

Use with WLST: Online

5.6.5.1 Description

Lists which types of instant messaging and presence servers Oracle WebCenter supports. Out-of-the-box, WebCenter applications support Oracle WebLogic Communication Server (OWLCS) and Microsoft Live Communication Server (LCS).

5.6.5.2 Syntax

```
listIMPAdapters()
```

5.6.5.3 Example

The following example lists which presence servers are supported.

```
wls:/weblogic/serverConfig> listIMPAdapters()
```

5.6.6 listIMPConnections

Module: Oracle WebCenter

Use with WLST: Online

5.6.6.1 Description

Lists all of the instant messaging and presence server connections that are configured for a named WebCenter application.

5.6.6.2 Syntax

```
listIMPConnections(appName, [name], [verbose], [server, applicationVersion)
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Optional. Name of an existing presence server connection. Use this argument to view connection details for a specific presence server connection. Note that if you use the `name` argument when `verbose` argument set to `true`, the `verbose` argument is ignored.
verbose	Optional. Displays presence server connection details in verbose mode. Valid values are `true` and `false`. When set to `true`, `listIMPConnections` lists all of the presence server connections that are configured for a WebCenter application, along with their details. When set to `false`, only connection names are listed. This argument defaults to `false`. If you use the `name` argument when `verbose` is set to `true`, the `verbose` argument is ignored.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.6.3 Examples

The following example lists all of the instant messaging and presence server connections that are configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig> listIMPConnections(appName='webcenter')
```

The following example lists all of the instant messaging and presence server connections that are configured for the application in verbose mode.

```
wls:/weblogic/serverConfig> listIMPConnections(appName='webcenter', verbose=true)
```

The following example lists connection details for an instant messaging and presence server connections named `impConnection1`.

```
wls:/weblogic/serverConfig> listIMPConnections(appName='webcenter',
name='impConnection1')
```

5.6.7 listDefaultIMPConnection

Module: Oracle WebCenter

Use with WLST: Online

5.6.7.1 Description

Lists the connection that the Instant Messaging and Presence service is using, in a named WebCenter application. While you can register multiple presence server connections for a WebCenter application, the Instant Messaging and Presence service only uses one connection —the default (or active) connection.

If only one presence server connection is available, that connection is assumed to be the default connection.

5.6.7.2 Syntax

```
listDefaultIMPConnection(appName, verbose, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
verbose	Optional. Displays the default presence server connection in verbose mode, if available. Valid options are `true` and `false`. When set to `true`, the name and details of the presence server connection are listed. When set to `false`, only the connection name displays. This argument defaults to `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.

Argument	Definition
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.7.3 Example

The following example lists the name and details of the connection that the Instant Messaging and Presence service is using, in an application named `webcenter`.

```
wls:/weblogic/serverConfig>listDefaultIMPConnection(appName='webcenter',
verbose=true)
```

5.6.8 setDefaultIMPConnection

Module: Oracle WebCenter

Use with WLST: Online

5.6.8.1 Description

Specifies the *default* connection for the Instant Messaging and Presence service, in a named WebCenter application. While you can register multiple presence server connections with a WebCenter application, the Instant Messaging and Presence service only uses one connection — the default (or active) connection.

If only one presence server connection is available, that connection is assumed to be the default connection.

5.6.8.2 Syntax

```
setDefaultIMPConnection(appName, name, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing instant messaging and presence connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.8.3 Example

The following example makes a connection named `myPresenceServer` the default (or active) connection for the Instant Messaging and Presence service.

```
wls:/weblogic/serverConfig>setDefaultIMPConnection(appName='webcenter',
name='myPresenceServer')
```

5.6.9 setIMPServiceProperty

Module: Oracle WebCenter

Use with WLST: Online

5.6.9.1 Description

Specifies default values for the Instant Messaging and Presence service.

Configurable properties for the Instant Messaging and Presence service are listed in Table 5–11, " Instant Messaging and Presence Service Configuration Properties".

Table 5–11 Instant Messaging and Presence Service Configuration Properties

Configuration Property	Description
selected.connection	Connection used by the Instant Messaging and Presence service.
rtc.cache.time	Cache timeout for instant messaging and presence data. The default is 60 seconds.
resolve.display.name.from .user.profile	Determines what to display if user display names are missing. When set to `false`, and display name information is unavailable, only the user name displays in the application. When set to `true`, and display name information is unavailable, display names are read from user profile data. Setting this option to `true` will impact performance. The default setting is false. Display names are not mandatory in presence data. If the WebCenter application does not always provide display names by default and you consider this information important, set `resolve.display.name.from.user.profile` to `true` so that display names always display.

5.6.9.2 Syntax

```
setIMPServiceProperty(appName, property, value, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
property	Name of the configuration property.
value	Value for the property.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.9.3 Example

The following example changes the default cache timeout for instant messaging and presence data, in an application named `webcenter`.

```
wls:/weblogic/serverConfig>setIMPServiceProperty(appName='webcenter',
property='rtc.cache.time', value='30')
```

5.6.10 removeIMPServiceProperty

Module: Oracle WebCenter

Use with WLST: Online

5.6.10.1 Description

Removes default value for the Instant Messaging and Presence service. Note that removing a default property might cause unexpected behavior.

> **Note:** Use this command syntax to disable the connection currently used by the Instant Messaging and Presence service:
>
> `removeIMPServiceProperty('appName='webcenter', 'property='selected.connection')`
>
> This command forces the `default` connection argument to `false`. See also, setIMPConnection.

5.6.10.2 Syntax

```
removeIMPServiceProperty(appName, property, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
property	Name of the configuration property.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.10.3 Example

The following example clears the default cache expiration value for the Instant Messaging and Presence service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig> removeIMPServiceProperty(appName='webcenter',
property='rtc.cache.time')
```

5.6.11 listIMPServiceProperties

Module: Oracle WebCenter

Use with WLST: Online

5.6.11.1 Description

Lists all configurable properties for the Instant Messaging and Presence service.

5.6.11.2 Syntax

```
listIMPServiceProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.6.11.3 Example

The following example lists configuration properties for the Instant Messaging and Presence service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig> listIMPServiceProperties(appName='webcenter')
```

5.7 Mail

Use the commands listed in Table 5–12 to manage mail server connections for a WebCenter application.

You can register multiple mail server connections:

- **WebCenter Spaces** supports multiple mail connections. The mail connection configured with `default=true` is the default connection for mail services in WebCenter Spaces. All additional connections are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences.
- **Custom WebCenter applications** only use one mail connection—the connection configured with `default=true`. Any additional connections are ignored.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 5–12 Mail WLST Commands

Use this command...	To...	Use with WLST...
createMailConnection	Create a mail server connection for a WebCenter application.	Online
setMailConnection	Edit an existing mail server connection.	Online
setMailConnectionProperty	Set mail server connection properties.	Online
deleteMailConnectionProperty	Delete a mail server connection property.	Online
listMailConnections	List all of the mail server connections that are configured for an application.	Online
listDefaultMailConnection	List the default mail server connection that is configured for an application.	Online
setDefaultMailConnection	Set a specified connection as the default mail server connection.	Online
setMailServiceProperty	Specify defaults for the Mail service.	Online

Table 5–12 (Cont.) Mail WLST Commands

Use this command...	To...	Use with WLST...
removeMailServiceProperty	Remove defaults for the Mail service.	Online
listMailServiceProperties	List Mail service properties.	Online

5.7.1 createMailConnection

Module: Oracle WebCenter

Use with WLST: Online

5.7.1.1 Description

Creates a mail server connection for a WebCenter application.

WebCenter applications support the Microsoft Exchange Server or any mail server that supports IMAP4 and SMTP. The most important mail server connection attributes are: `imapHost`, `imapPort`, `imapSecured`, `smtpHost`, `smtpPort`, and `smtpSecured`

You can register multiple mail server connections:

- **WebCenter Spaces** supports multiple mail connections. The mail connection configured with `default=true` is the default connection for mail services in WebCenter Spaces. All additional connections are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences.
- **Custom WebCenter applications** only use one mail connection—the connection configured with `default=true`. Any additional connections are ignored.

5.7.1.2 Syntax

```
createMailConnection(appName, name, imapHost, imapPort, smtpHost, smtpPort,
imapSecured, smtpSecured, appId, [timeout, default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.
imapHost	Host name of the machine on which the IMAP service is running.
imapPort	Port on which the IMAP service listens.
smtpHost	Host name of the machine where the SMTP service is running.
smtpPort	Port on which the SMTP service listens.
imapSecured	Optional. Specifies whether the mail server connection to the IMAP server is SSL-enabled. Valid values are `true` and `false`. The default for this argument is `false`.
smtpSecured	Optional. Specifies whether the SMTP server is secured. Valid values are `true` and `false`. The default for this argument is `false`.

Argument	Definition
appId	External application associated with the mail server connection. External application credential information is used to authenticate users against the IMAP and SMTP servers. The same credentials are supplied to authenticate the user on both the IMAP and SMTP servers. See also createExtAppConnection. The external application you configure for the Mail service must use `authMethod=POST`, and specify an additional field with `fieldName='Email Address'` and `displaytoUser=true`. See also addExtAppField and setExtAppField.
timeout	Optional. Length of time (in seconds) that the service waits to acquire a connection before terminating. This argument defaults to `-1`. When set to `-1`, the service default (10 seconds) applies.
default	Optional. Indicates whether this connection is the default connection for the Mail service. Valid values are `true` and `false`. This argument defaults to `false`. ▪ **WebCenter Spaces** supports multiple mail connections. The mail connection configured with `default=true` is the default connection for mail services in WebCenter Spaces. Additional connections, configured with `default=false`, are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences. ▪ **Custom WebCenter applications** only use one mail connection—the connection configured with `default=true`. Any additional connections are ignored.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.1.3 Examples

The following example creates a mail server connection named `myMailConnection` and enables the mail server connection in the WebCenter application.

```
wls:/weblogic/serverConfig> createMailConnection(appName='webcenter' ,
name='myMailConnection' , imapHost='myimaphost.com', imapPort=143 ,
smtpHost='mysmtphost.com' , smtpPort=25 , imapSecured=false, smtpSecured=false,
appId='extApp_Mail', timeout=60, default=true)
```

5.7.2 setMailConnection

Module: Oracle WebCenter

Use with WLST: Online

5.7.2.1 Description

Edits an existing mail connection. Use this command to update connection attributes.

The connection is created using the createMailConnection command.

(WebCenter Spaces application only.) This command enables you to set additional, optional, LDAP server attributes that cannot be set using `createMailConnection`. When LDAP details are defined, the Mail service creates, edits, and deletes group

space distribution lists for WebCenter Spaces. group space distribution lists are named after their group space (excluding non-java identifiers) and assigned a domain (derived from the `domain` attribute, for example, `@mycompany.com`). If LDAP details are not provided, group space distribution lists are not created or maintained. The mail server must be a *Microsoft Exchange Server*.

5.7.2.2 Syntax

```
setMailConnection(appName, name, [imapHost, imapPort, smtpHost, smtpPort,
imapSecured, smtpSecured, appId, default, ldapHost, ldapPort, ldapBaseDN,
ldapAdminUser, ldapAdminPassword, ldapSecured, domain, defaultUser, timeout,
server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing mail server connection.
imapHost	Optional. Host name of the machine on which the IMAP service is running.
imapPort	Optional. Port on which the IMAP service listens.
smtpHost	Optional. Host name of the machine where the SMTP service is running.
smtpPort	Optional. Port on which the SMTP service listens.
imapSecured	Optional. Specifies whether the connection to the IMAP server is secured (SSL-enabled). Valid values are `true` and `false`. The default for this argument is `false`.
smtpSecured	Optional. Specifies whether the connection to the SMTP server is secured (SSL-enabled). Valid values are `true` and `false`. The default for this argument is `false`.
appId	Optional. External application associated with the mail server connection. External application credential information is used to authenticate users against the IMAP and SMTP servers. The same credentials are supplied to authenticate the user on both the IMAP and SMTP servers. See also createExtAppConnection. The external application you configure for the Mail service must use `authMethod=POST`, and specify an additional field with `fieldName='Email Address'` and `displaytoUser=true`. See also addExtAppField and setExtAppField.
ldapHost	Optional. Host name of the machine where the LDAP directory server is running.
ldapPort	Optional. Port on which the LDAP directory server listens.
ldapBaseDN	Optional. Base distinguished name for the LDAP schema. For example, `CN=Users,DC=oracle,DC=com`.
ldapAdminUser	Optional. User name of the LDAP directory server administrator. A valid administrator with privileges to make entries into the LDAP schema.
ldapAdminPassword	Optional. Password for the LDAP directory server administrator. This password will be stored in a secured store.

Argument	Definition
ldapSecured	Optional. Specifies whether the connection to the LDAP server is secured (SSL enabled). Valid values are `true` and `false`. The default for this argument is `false`. Set this to `true` for all LDAP communications over SSL.
domain	Optional. Domain name appended to group space distribution lists. For example, if the domain attribute is set to `mycompany.com`, the `Finance Project` group space will maintain a distribution list named `FinanceProject@oracle.com`.
defaultUser	Optional. Comma-delimited list of user names to whom you want to grant moderation capabilities. These users become members of every group space distribution list that is created. The users specified must exist in the Base LDAP schema (specified in the `ldapBaseDN` argument).
timeout	Optional. Length of time (in seconds) that the service waits to acquire a connection before terminating. This argument defaults to `-1`. When set to `-1`, the service default (10 seconds) applies.
default	Optional. Indicates whether this connection is the default (or active) connection for the Mail service. Valid values are `true` and `false`. This argument defaults to `false`. `true` specifies that this connection is the default connection for the Mail service. ▪ **WebCenter Spaces** supports multiple mail connections. The mail connection configured with `default=true` is the default connection for mail services in WebCenter Spaces. Additional connections, configured with `default=false`, are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences. ▪ **Custom WebCenter applications** only use one mail connection—the connection configured with `default=true`. Any additional connections are ignored. A connection does not cease to be the default connection for the Mail service if you change the `default` value from `false` to `true`. To stop using a default connection, use the removeMailServiceProperty command as follows: `removeMailServiceProperty('appName='webcenter', 'property='selected.connection')`
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.2.3 Examples

The following example sets individual attributes of a mail server connection.

```
wls:/weblogic/serverConfig>setMailConnection(appName='webcenter',
name='myMailConnection', imapHost='myimaphost.com', imapPort=143,
smtpHost='mysmtphost.com' , smtpPort=25 , imapSecured=false, smtpSecured=false,
appId='extApp_Mail', timeout=60, default=true)
```

The following example sets individual attributes of a mail server connection.

```
wls:/weblogic/serverConfig>setMailConnection(appName='webcenter',
name='myMailConnection', imapPort=993, imapSecured=true, smtpPort=465 ,
smtpSecured=true)
```

The following example sets LDAP attributes for a mail server connection.

```
wls:/weblogic/serverConfig>setMailConnection(appName='webcenter',
name='myMailConnection', domain='ORACLE.COM', defaultUser='admin',
imapHost='myimaphost.com', imapPort=143, smtpHost='mysmtphost.com',
imapSecured=false, smtpSecured=false, smtpPort=25, appId='extApp_Mail',
default=true, ldapHost='myldaphost.com', ldapPort=389,
ldapBaseDN='CN=Users,DC=exchange,DC=uk,DC=com', ldapAdminUser='administrator',
ldapAdminPassword='adminpswd', ldapSecured=false, timeout=60)
```

5.7.3 setMailConnectionProperty

Module: Oracle WebCenter

Use with WLST: Online

5.7.3.1 Description

Sets a mail server connection property. Use this command if additional parameters are required to connect to your mail server. This is an extensible way to add any connection property using a key and a value. (You are not limited to connection properties specified by createMailConnection and setMailConnection.)

All known, additional connection properties are listed in Table 5–13, " Additional Mail Connection Properties".

Table 5–13 Additional Mail Connection Properties

Additional Connection Property	Description
`charset`	Characterset used on the connection. The default charset is `UTF-8`. To use a different characterset, such as `ISO-8859-1`, set the `charset` connection property.
Various IMAP properties	Any valid IMAP connection property. For example, `mail.imap.connectionpoolsize`. A list of valid IMAP properties are available at: `http://java.sun.com/products/javamail/javadocs/com/sun/mail/imap/package-summary.html`
Various SMTP properties	Any valid SMTP connection property. For example, `mail.smtp.timeout`. A list of valid SMTP properties are available at: `http://java.sun.com/products/javamail/javadocs/com/sun/mail/smtp/package-summary.html`

Note: Do not use the setMailConnectionProperty to set connection properties available through createMailConnection or setMailConnection. Attempting to do so, has no effect.

5.7.3.2 Syntax

```
setMailConnectionProperty(appName, name, key, value, [secure], [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing mail server connection.
key	Name of the connection property.
value	Value for the property. Allows any property to be modified on the connection with a key and value.
secure	Optional. Indicates whether the property value must be stored securely using encryption. Valid options are `true` and `false`. When `true`, the value is encrypted. The default option is `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.3.3 Example

The following example adds a custom mail server connection property called `myProperty1` with a default value `propertyValue1`.

```
wls:/weblogic/serverConfig> setMailConnectionProperty(appName='webcenter',
name='myMailServer', key='myProperty1', value='propertyValue1')
```

5.7.4 deleteMailConnectionProperty

Module: Oracle WebCenter

Use with WLST: Online

5.7.4.1 Description

Deletes a mail server connection property. Take care when deleting connection properties because the connection may not work as expected if the configuration becomes invalid as a result.

This command can only delete *additional* connection properties added using the `setMailConnectionProperty` command.

5.7.4.2 Syntax

```
deleteMailConnectionProperty(appName, name, key, [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing mail server connection.
key	Name of the connection property you want to delete.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.

Argument	Definition
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.4.3 Example

The following example deletes a mail server connection property named `mailProperty1`.

```
wls:/weblogic/serverConfig> deleteMailConnectionProperty(appName='webcenter',
name='myMailServer', key='mailProperty1')
```

5.7.5 listMailConnections

Module: Oracle WebCenter

Use with WLST: Online

5.7.5.1 Description

Lists all of the mail server connections that are configured for a named WebCenter application.

5.7.5.2 Syntax

```
listMailConnection(appName, [name, verbose, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Optional. Name of an existing mail server connection. Use this argument to view connection details for a specific mail server connection.
verbose	Optional. Displays mail server connection details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listMailConnections` lists all of the mail server connections that are configured for a WebCenter application, along with their details. When set to `false`, only connection names are listed. This argument defaults to `false`. If you use the `name` argument when `verbose` is set to `true`, the `verbose` argument is ignored.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.5.3 Example

The following example lists the names of mail server connections that are currently configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig> listMailConnections(appName='webcenter')
```

The following example lists connection names and details for all of the mail server connections that are currently configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig> listMailConnections(appName='webcenter', verbose=true)
```

The following example lists connection details for a mail server connection named `mailConnection1`.

```
wls:/weblogic/serverConfig> listMailConnections(appName='webcenter',
name='mailConnection1')
```

5.7.6 listDefaultMailConnection

Module: Oracle WebCenter

Use with WLST: Online

5.7.6.1 Description

Lists the default mail server connection that the Mail service is using, in a named WebCenter application.

You can register multiple mail server connections but there can only be one default connection:

- **WebCenter Spaces** supports multiple mail connections. The mail connection configured with `default=true` is the default connection for mail services in WebCenter Spaces. All additional connections are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences.
- **Custom WebCenter applications** only use one mail connection—the connection configured with `default=true`. Any additional connections are ignored.

5.7.6.2 Syntax

```
listDefaultMailConnection(appName,[verbose], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
verbose	Optional. Displays the default mail server connection in verbose mode, if available. Valid options are `true` and `false`. When set to `true`, the name and details of the mail server connection are listed. When set to `false`, only the connection name displays. This argument defaults to `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.6.3 Example

The following example lists the name and details of the mail server connection that the Mail service is using, in an application named `webcenter`.

```
wls:/weblogic/serverConfig> listDefaultMailConnection(appName='webcenter',
verbose=true)
```

5.7.7 setDefaultMailConnection

Module: Oracle WebCenter

Use with WLST: Online

5.7.7.1 Description

Specifies the *default* mail server connection for the Mail service, in a named WebCenter application.

You can register multiple mail server connections but there can only be one default connection:

- **WebCenter Spaces** supports multiple mail connections. The mail connection configured with `default=true` is the default connection for mail services in WebCenter Spaces. All additional connections are offered as alternatives; WebCenter Spaces users can choose which one they want to use through user preferences.
- **Custom WebCenter applications** only use one mail connection—the connection configured with `default=true`. Any additional connections are ignored.

5.7.7.2 Syntax

```
setDefaultMailConnection(appName, name, [server], [applicationVersion])
```

Argument	Description
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing mail connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.7.3 Example

The following example configures the Mail service to use a connection named `myMailServer`.

```
wls:/weblogic/serverConfig>setDefaultMailConnection(appName='webcenter',
name='myMailServer')
```

5.7.8 setMailServiceProperty

Module: Oracle WebCenter

Use with WLST: Online

5.7.8.1 Description

Specifies default values for the Mail service.

Configurable properties for the Mail service are listed in Table 5–14, " Mail Service Configuration Properties".

Table 5–14 Mail Service Configuration Properties

Configuration Property	Description
mail.messages.fetch.size	Maximum number of messages displayed in mail inboxes.

5.7.8.2 Syntax

```
setMailServiceProperty(appName, property, value, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
property	Name of the configuration property
value	Value for the property.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.8.3 Example

The following example increases the default number of messages displayed in mail inboxes to 100, in an application named `webcenter`.

```
wls:/weblogic/serverConfig>setMailServiceProperty(appName='webcenter',
property='mail.messages.fetch.size', value='100')
```

5.7.9 removeMailServiceProperty

Module: Oracle WebCenter

Use with WLST: Online

5.7.9.1 Description

Removes default values for the Mail service. Note that removing a default property might cause unexpected behavior.

Note: Use this command syntax to stop the Mail service from using the current default connection:

```
removeMailServiceProperty('appName='webcenter',
'property='selected.connection')
```

This command forces the `default` connection argument to `false`. See also, setMailConnection.

5.7.9.2 Syntax

```
removeMailServiceProperty(appName, property, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
property	Name of the configuration property.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.9.3 Example

The following example clears the current `mail.messages.fetch.size` setting for the Mail service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig>removeMailServiceProperty(appName='webcenter',
property='mail.messages.fetch.size')
```

5.7.10 listMailServiceProperties

Module: Oracle WebCenter

Use with WLST: Online

5.7.10.1 Description

Lists all configurable properties for the Mail service.

5.7.10.2 Syntax

```
listMailServiceProperties(appName, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.7.10.3 Example

The following example lists configuration properties for the Mail service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig>listMailServiceProperties(appName='webcenter')
```

5.8 Producer

Use the commands listed in Table 5–15 to manage portlet producers used in WebCenter applications.

All configuration changes made using these WebCenter WLST commands are immediately available in the WebCenter application.

Table 5–15 Producer WLST Commands

Use this command...	To...	Use with WLST...
registerWSRPProducer	Create and register a WSRP producer.	Online
setWSRPProducer	Edit WSRP producer registration details.	Online
listWSRPProducers	List WSRP producer registration details.	Online
deregisterWSRPProducer	Deregister a WSRP producer, and delete the associated WSRP and Web Service connections.	Online
listWSRPProducerRegistrationProperties	List registration properties supported by a WSRP producer.	Online
listWSRPProducerUserCategories	List any user categories that the WSRP producer might support.	Online
mapWSRPProducerUserCategory	Map a role that is defined in the specified application to a user category supported by a WSRP producer.	Online
registerPDKJavaProducer	Create and register an Oracle PDK-Java producer.	Online
setPDKJavaProducer	Edit PDK-Java producer registration details.	Online
deregisterPDKJavaProducer	Deregister an Oracle PDK-Java producer, deleting the associated connection.	Online
listPDKJavaProducers	List registered Oracle PDK-Java producers.	Online
refreshProducer	Refresh the metadata stored for the named producer to reflect the portlets currently offered by that producer.	Online
registerOOTBProducers	Register out-of-the-box producers provided with Oracle WebCenter.	Online
deregisterOOTBProducers	Deregister out-of-the-box producers provided with Oracle WebCenter.	Online
registerSampleProducers	Register the sample producers provided with Oracle WebCenter.	Online
deregisterSampleProducers	Deregister sample producers.	Online

5.8.1 registerWSRPProducer

Module: Oracle WebCenter

Use with WLST: Online

5.8.1.1 Description

Creates a connection to a WSRP portlet producer and registers the WRSP producer with a named WebCenter application. When you create a WSRP producer connection, a Web Service connection is also created named `<name>-wsconn` where `<name>` is the value specified for the name argument.

5.8.1.2 Syntax

```
registerWSRPProducer(appName, name, url, [proxyHost], [proxyPort],
[timeout],[externalApp],[registrationProperties],[tokenType],[issuer],[defUser],
```

```
[keyStorePath],[keyStorePswd],[sigKeyAlias],[sigKeyPswd],[encKeyAlias],
[encKeyPswd],[recptAlias],[server],[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application. The name you specify here will appear in the Oracle Composer (under the *Portlets* folder).

url	Producer WSDL URL. The syntax will vary according to your WSRP implementation, for example:
	`http://host_name:port_number/context_root/portlets/wsrp2?WSDL`
	`http://host_name:port_number/context_root/portlets/wsrp1?WSDL`
	`http://host_name:port_number/context_root/portlets/?WSDL` (WSRP 1.0 for backward compatibility)
	Where:
	▪ `host_name` is the server where your producer is deployed
	▪ `port_number` is the HTTP listener port number
	▪ `context_root` is the Web application's context root
	▪ `portlets[/wsrp(1|2)]?WSDL` is static text. The text entered here depends on how the producer is deployed.
	For example:
	`http://myhost.com:7778/MyPortletApp/portlets/wsrp2?WSDL`

proxyHost	Optional. Host name or IP address of the proxy server. A proxy is required when the WebCenter application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed to communicate with the producer.
proxyPort	Optional. Port number on which the proxy server listens.
timeout	Optional. Timeout setting for communications with the producer, in seconds. For example, the maximum time the producer may take to register, deregister, or display portlets on WebCenter pages. This argument defaults to `30`. Individual portlets may define their own timeout period, which takes precedence over the value expressed here.
registrationProperties	Optional. A list of registration properties and their values. The format of this argument must be a comma-separated list of valid registration properties, each followed by an equals symbol and the value. For example: `name=Producer,key=123`. The registration properties for a producer can be found using `listWSRPProducerRegistrationProperties`. See Section 5.8.5, "listWSRPProducerRegistrationProperties".

Argument	Definition
tokenType	Optional. Type of token profile to use for authentication with this WSRP producer. Valid values are: ■ **USERNAME_WITHOUT_PASSWORD** (oracle/wss10_username_id_propagation_with_msg_protection_client_policy)—Enforces message level protection (integrity and confidentiality) and identity propagation for inbound SOAP requests using mechanisms described in WS-Security 1.0. Message protection is provided using WS-Security 1.0's Basic128 suite of asymmetric key technologies. Specifically RSA key mechanisms for confidentiality, SHA-1 hashing algorithm for integrity and AES-128 bit encryption. Identity is set using a *username* provided through the UsernameToken WS-Security SOAP header. The subject is established against the currently configured identity store. ■ **USERNAME_WITH_PASSWORD** (oracle/wss10_username_token_with_message_protection_client_policy)—Enforces message-level protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security v1.0 standard. It uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. Authentication is enforced using *credentials* in the WS-Security UsernameToken SOAP header. The subject is established against the currently configured identity store. Use this token profile if the WSRP producer has a different identity store. You will need to define an external application pertaining to the producer and associate the external application with this producer. ■ **SAML_TOKEN_WITH_MSG_INTEGRITY** (wss10_saml_token_with_message_integrity_client_policy)—Enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. It uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity. When this policy is selected, the recipient key alias (recptAlias) should be disabled. ■ **SAML_TOKEN_WITH_MSG_PROTECTION** (oracle/wss10_saml_token_with_message_protection_client_policy)—Enforces message-level protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. It uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption.
issuer	Optional. Name of the issuer of the token. The issuer name is the entity that vouches for the verification of the subject. This argument only applies when the tokenType is `SAML_TOKEN_WITH_MSG_PROTECTION` or `SAML_TOKEN_WITH_MSG_INTEGRITY`.
defUser	Optional. User name to assert to the remote producer when the user is not authenticated with the WebCenter application. This argument applies when the `tokenType` is `SAML_TOKEN_WITH_MSG_PROTECTION`, `SAML_TOKEN_WITH_MSG_INTEGRITY`, or `USERNAME_WITH_PASSWORD`.

Argument	Definition
extApp	Optional. This argument applies when the tokenType is USERNAME_WITH_PASSWORD. If this producer uses an external application to store and supply user credentials for authentication, use this argument to name the associated external application.
keyStorePath	Optional. Full path to the key store that contains the certificate and the private key that is used for signing some parts of the SOAP message, such as the security token and SOAP message body. The selected file should be a key store created with the Java keytool.
keyStorePswd	Optional. Password to the key store that was set when the key store was created.
sigKeyAlias	Optional. Identifier for the certificate associated with the private key that is used for signing. A valid value is one of the key aliases that is located in the specified key store.
sigKeyPswd	Optional. Password for accessing the key identified by the alias that is specified using the sigKeyAlias argument.
encKeyAlias	Optional. Key alias to be used for encryption. A valid value is one of the key aliases that is located in the specified key store.
encKeyPswd	Optional. Password for accessing the encryption key.
recptAlias	Optional. Key store alias that is associated with the producer's certificate. This certificate is used to encrypt the message to the producer. Do not specify a recipient key alias when the tokenType is SAML_TOKEN_WITH_MSG_INTEGRITY.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.1.3 Examples

The following example registers a WSRP producer named WSRPSamples and registers the WSRP producer with an application named webcenter.

```
wls:/weblogic/serverConfig> registerWSRPProducer(appName='webcenter',
name='WSRPSamples', url='http://myhost.com:9999/
portletapp/portlets/wsrp2?WSDL')
```

The following example registers a secure WSRP producer.

```
wls:/weblogic/serverConfig> registerWSRPProducer(appName='webcenter',
name='WSRPSamples2', url='http://myhost.com:8899/portletapp/portlets/wsrp2?WSDL',
tokenType='USERNAME_WITHOUT_PASSWORD', issuer='SMITH', defUser='Smith',
keyStorePath='/keys/mykeystore.jks', keyStorePswd='Test1',
sigKeyAlias='mysigalias',sigKeyPswd='mysigpswd', encKeyAlias='myencalias',
encKeyPswd='myencpswd', recptAlias='myrcptalias')
```

5.8.2 setWSRPProducer

Module: Oracle WebCenter

Use with WLST: Online

5.8.2.1 Description

Edits registration details for an existing WSRP producer.

5.8.2.2 Syntax

```
setWSRPProducer(appName, name, url, [proxyHost, [proxyPort]], [timeout],
[externalApp], [tokenType],[issuer], [defUser], [keyStorePath], [keyStorePswd]
[sigKeyAlias], [sigKeyPswd], [encKeyAlias], [encKeyPswd], [recptAlias], [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing WSRP producer.

url	Optional. WSRP producer URL. The syntax will vary according to your WSRP implementation, for example:
	`http://host_name:port_number/context_root/portlets/wsrp2?WSDL`
	`http://host_name:port_number/context_root/portlets/wsrp1?WSDL`
	`http://host_name:port_number/context_root/portlets/?WSDL` (WSRP 1.0 for backward compatibility)
	Where:
	■ `host_name` is the server where your producer is deployed
	■ `port_number` is the HTTP listener port number
	■ `context_root` is the Web application's context root
	■ `portlets[/wsrp(1|2)]?WSDL` is static text. The text entered here depends on how the producer is deployed.
	For example:
	`http://myhost:7778/MyPortletApp/portlets/wsrp2?WSDL`

proxyHost	Optional. Host name or IP address of the proxy server. A proxy is required when the WebCenter application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed to communicate with the producer.
proxyPort	Optional. Port number on which the proxy server listens.
timeout	Optional. Timeout setting for communications with the producer, in seconds. For example, the maximum time the producer may take to register, deregister, or display portlets on WebCenter pages. This argument defaults to `30`. Individual portlets may define their own timeout period, which takes precedence over the value expressed here.
extApp	Optional. This argument applies when the `tokenType` is `USERNAME_WITH_PASSWORD`. If this producer uses an external application to store and supply user credentials for authentication, use this argument to name the associated external application.

Argument	Definition
tokenType	Optional. Type of token profile to use for authentication with this WSRP producer. Valid values are: ■ **USERNAME_WITHOUT_PASSWORD** (oracle/wss10_username_id_propagation_with_msg_protection_client_policy)—Enforces message level protection (integrity and confidentiality) and identity propagation for inbound SOAP requests using mechanisms described in WS-Security 1.0. Message protection is provided using WS-Security 1.0's Basic128 suite of asymmetric key technologies. Specifically RSA key mechanisms for confidentiality, SHA-1 hashing algorithm for integrity and AES-128 bit encryption. Identity is set using a *username* provided through the UsernameToken WS-Security SOAP header. The subject is established against the currently configured identity store. ■ **USERNAME_WITH_PASSWORD** (oracle/wss10_username_token_with_message_protection_client_policy)—Enforces message-level protection (message integrity and confidentiality) and authentication for inbound SOAP requests in accordance with the WS-Security v1.0 standard. It uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanism for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. Authentication is enforced using *credentials* in the WS-Security UsernameToken SOAP header. The subject is established against the currently configured identity store. Use this token profile if the WSRP producer has a different identity store. You will need to define an external application pertaining to the producer and associate the external application with this producer. ■ **SAML_TOKEN_WITH_MSG_INTEGRITY** (wss10_saml_token_with_message_integrity_client_policy)—Enforces message-level integrity protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. It uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity. When this policy is selected, the recipient key alias (recptAlias) should be disabled. ■ **SAML_TOKEN_WITH_MSG_PROTECTION** (oracle/wss10_saml_token_with_message_protection_client_policy)—Enforces message-level protection and SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.0 standard. It uses WS-Security's Basic 128 suite of asymmetric key technologies, specifically RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption.
issuer	Optional. Name of the issuer of the token. The issuer name is the entity that vouches for the verification of the subject. This argument only applies when the tokenType is `SAML_TOKEN_WITH_MSG_PROTECTION` or `SAML_TOKEN_WITH_MSG_INTEGRITY`.
defUser	Optional. User name to assert to the remote producer when the user is not authenticated with the WebCenter application. This argument applies when the `tokenType` is `SAML_TOKEN_WITH_MSG_PROTECTION`, `SAML_TOKEN_WITH_MSG_INTEGRITY`, or `USERNAME_WITH_PASSWORD`.

Argument	Definition
keyStorePath	Optional. Full path to the key store that contains the certificate and the private key that is used for signing some parts of the SOAP message, such as the security token and SOAP message body. The selected file should be a key store created with the Java keytool.
keyStorePswd	Optional. Password to the key store that was set when the key store was created.
sigKeyAlias	Optional. Identifier for the certificate associated with the private key that is used for signing. A valid value is one of the key aliases that is located in the specified key store.
sigKeyPswd	Optional. Password for accessing the key identified by the alias that is specified using the `sigKeyAlias` argument.
encKeyAlias	Optional. Key alias to be used for encryption. A valid value is one of the key aliases that is located in the specified key store.
encKeyPswd	Optional. Password for accessing the encryption key.
recptAlias	Optional. Key store alias that is associated with the producer's certificate. This certificate is used to encrypt the message to the producer. Do not specify a recipient key alias when the `tokenType` is `SAML_TOKEN_WITH_MSG_INTEGRITY`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.2.3 Example

This example increases the timeout, for the `WSRPSamples` producer, to 60 seconds.

```
wls:/weblogic/serverConfig>setWSRPProducer(appName='webcenter',
name='WSRPSamples', timeout=60)
```

This example updates security properties on a secure WSRP producer.

```
wls:/weblogic/serverConfig>setWSRPProducer(appName='webcenter',
name='WSRPSamples2', tokenType='USERNAME_WITHOUT_PASSWORD', issuer='SMITH',
defUser='Smith', keyStorePath='/keys/mykeystore.jks', keyStorePswd='Test1',
sigKeyAlias='mysigalias', sigKeyPswd='mysigpswd',encKeyAlias='myencalias',
encKeyPswd='myencpswd', recptAlias='myrcptalias')
```

This example removes all the security properties set on a secure WSRP producer.

```
wls:/weblogic/serverConfig>setWSRPProducer(appName='webcenter',
name='WSRPSamples2', tokenType='')
```

5.8.3 listWSRPProducers

Module: Oracle WebCenter

Use with WLST: Online

5.8.3.1 Description

Lists WSRP producer registration details.

5.8.3.2 Syntax

```
listWSRPProducers(appName,[name],[verbose], [server], [applicationVersion])
```

Argument	Definition
appName	The name of the application in which one or more WSRP producers is registered.
name	Optional. Name of an existing WSRP producer. If omitted, connection details for all WSRP producers configured for this WebCenter application are listed.
verbose	Optional. Displays WSRP producer connection details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listWSRPProducers` lists all connection properties. When set to `false`, `listWSRPProducers` lists connection names only. This argument defaults to `true`. If you set this argument to `false`, do not specify the `names` argument.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.3.3 Example

```
wls:/weblogic/serverConfig> listWSRPProducers(appName='webcenter',
name='WSRPSamples')
```

5.8.4 deregisterWSRPProducer

Module: Oracle WebCenter

Use with WLST: Online

5.8.4.1 Description

Deregisters a WSRP producer, and deletes the associated WSRP and Web Service connections.

5.8.4.2 Syntax

```
deregisterWSRPProducer(appName, name, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application where the producer is registered.
name	Name of an existing WSRP producer.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.4.3 Example

The following example deregisters the `WSRPSamples` producer in an application named `webcenter`.

```
wls:/weblogic/serverConfig> deregisterWSRPProducer(appName='webcenter',
name='WSRPSamples')
```

5.8.5 listWSRPProducerRegistrationProperties

Module: Oracle WebCenter

Use with WLST: Online

5.8.5.1 Description

Lists registration properties supported by a WSRP portlet producer.

5.8.5.2 Syntax

```
listWSRPProducerRegistrationProperties(appName, url,[proxyHost, [proxyPort],
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.

url	WSRP producer URL. The syntax will vary according to your WSRP implementation, for example:
	`http://host_name:port_number/context_root/portlets/wsrp2?WSDL`
	`http://host_name:port_number/context_root/portlets/wsrp1?WSDL`
	`http://host_name:port_number/context_root/portlets/?WSDL` (WSRP 1.0 for backward compatibility)
	Where:
	■ `host_name` is the server where your producer is deployed
	■ `port_number` is the HTTP listener port number
	■ `context_root` is the Web application's context root
	■ `portlets[/wsrp(1|2)]?WSDL` is static text. The text entered here depends on how the producer is deployed.
	For example:
	`http://myhost:7778/MyPortletApp/portlets/wsrp2?WSDL`

proxyHost	Optional. Host name or IP address of the proxy server. A proxy is required when the WebCenter application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed to communicate with the producer.
proxyPort	Optional. Port number on which the proxy server listens.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.5.3 Example

The following example lists valid registration properties for the WSRP producer with the WSDL URL provided.

```
wls:/weblogic/serverConfig> listWSRPProducerRegistrationProperties
(appName='webcenter', url='http://myhost:9999/portletapp/portlets/wsrp2?WSDL')
Registration Property hint : hint text
Registration Property label : label text
Registration Property language : en
Registration Property name : {urn:xyz:wlp:prop:reg:registration}consumerRole
Registration Property value : None
```

5.8.6 listWSRPProducerUserCategories

Module: Oracle WebCenter

Use with WLST: Online

5.8.6.1 Description

Lists any user categories that a WSRP producer might support. WebCenter users can use the WLST command mapWSRPProducerUserCategory to map application roles to a producer's user category.

5.8.6.2 Syntax

```
listWSRPProducerUserCategories(appName, name, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing WSRP producer.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.6.3 Example

The following example displays the categories associated with a WSRP producer named `WSRPSamples`.

```
wls:/weblogic/serverConfig> listWSRPProducerUserCategories(appName='webcenter',
name='WSRPSamples')
User Category Name : categoryTwo
User Category Description : Custom role two.
User Category Mapped Local Roles : None


User Category Name : categoryOne
User Category Description : Custom role one.
User Category Mapped Local Roles : None
```

5.8.7 mapWSRPProducerUserCategory

Module: Oracle WebCenter

Use with WLST: Online

5.8.7.1 Description

Maps a role that is defined in the specified WebCenter application to a user category supported by a WSRP producer. The user categories may be found using listWSRPProducerUserCategories.

5.8.7.2 Syntax

```
mapWSRPProducerUserCategory(appName, name, localRole, producerUserCategory,
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing WSRP producer.
localRole	Name of the WebCenter application role to be mapped.
producerUserCategory	WSRP producer user category to which the WebCenter role will be mapped.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.7.3 Example

The following example maps the application role `admin` to the WSRP user category `wrsp-admin`.

```
wls:/weblogic/serverConfig> mapWSRPProducerUserCategory(appName='webcenter',
name='WSRPProducer1', localRole='admin', producerUserCategory='wsrp-admin')
```

5.8.8 registerPDKJavaProducer

Module: Oracle WebCenter

Use with WLST: Online

5.8.8.1 Description

Creates a connection to an Oracle PDK-Java portlet producer and registers the Oracle PDK-Java producer with a named WebCenter application.

5.8.8.2 Syntax

```
registerPDKJavaProducer(appName, name, url, [serviceId], [proxyHost,
[proxyPort]], [subscriberId], [sharedKey], [timeout],
[establishSession],[externalApp], [mapUser], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application for which you want to perform this operation.

Argument	Definition
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.

url	URL for the Oracle PDK-Java producer. Use the following syntax:
	`http://host_name:port_number/context_root/providers`
	Where:
	■ `host_name` is the server where the producer is deployed
	■ `port_number` is the HTTP Listener port number
	■ `context_root` is the Web application's context root.
	■ `providers` is static text. The text entered here depends on how the producer is deployed.
	For example:
	`http://myHost:7778/myEnterprisePortlets/providers`

serviceId	Optional. Service ID of the producer.
	PDK-Java enables you to deploy multiple producers under a single adapter servlet. Producers are identified by their unique service ID. A service ID is required only if the service ID is not appended to the URL end point.
	For example, the following URL endpoint requires `sample` as the service ID:
	`http://domain.us.oracle.com:7778/axyz/providers`
	However, the following URL endpoint, does not require a service ID:
	`http://domain.us.oracle.com:7778/axyz/providers/sample`
	The service ID is used to look up a file called `<service_id>.properties`, which defines the characteristics of the producer, such as whether to display its test page. Use any value to create the service ID.

proxyHost	Optional. Host name or IP address of the proxy server. A proxy is required if the WebCenter application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed for communication with the producer.
proxyPort	Optional. Port number on which the proxy server listens. This argument defaults to `80`.
sharedKey	Optional. Shared key used for message authentication with the remote producer. Message authentication ensures that the incoming messages are sent from a host with a shared key. This argument defaults to null. The shared key can contain between 10 and 20 alphanumeric characters.
subscriberId	Optional. Consumer's identifier, if required. When a producer is registered with an application, a call is made to the producer. During the call, the consumer (WebCenter application in this instance) passes the value for Subscriber ID to the producer. The producer may be coded to use the subscriber ID.

Argument	Definition
timeout	Optional. Timeout setting for communications with the producer, in seconds. For example, the maximum time the producer may take to register, deregister, or display portlets on WebCenter pages. This argument defaults to `30`. Individual portlets may define their own timeout period, which takes precedence over the value expressed here.
establishSession	Optional. Enable a user session when executing portlets from this producer. Valid values are `true` and `false`. The default for this argument is `false`. When sessions are enabled (`true`), the server maintains session-specific information, such as the user name. Message authentication uses sessions, so if a shared key is specified, this option should also be enabled. For sessionless communication between the producer and the server, specify `false`.
externalApp	Optional. Name of the external application with which to associate the producer. Required if one of this producer's portlets requires authentication.
mapUser	Optional. Flag indicating whether the mapped username from the external application should be passed to the producer. Valid values are `true` and `false`. This argument defaults to `true`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.8.3 Example

The following example creates and registers an Oracle PDK-Java producer named `JPDKSamples`, for an application named `webcenter`.

```
wls:/weblogic/serverConfig> registerPDKJavaProducer(appName='webcenter',
name='JPDKSamples', url='http://myhost:9999/jpdk/providers/sample')
```

5.8.9 setPDKJavaProducer

Module: Oracle WebCenter

Use with WLST: Online

5.8.9.1 Description

Edits registration details for an existing PDK-Java producer.

5.8.9.2 Syntax

```
setPDKJavaProducer(appName, name, url, [serviceId], [proxyHost, [proxyPort]],
[subscriberId], [sharedKey], [timeout], [establishSession], [externalApp],
[mapUser], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.

Argument	Definition
name	Name of an existing PDK-Java producer.

url	URL for the Oracle PDK-Java producer. Use the following syntax:
	`http://`*`host_name`*`:`*`port_number`*`/`*`context_root`*`/providers`
	Where:
	■ `host_name` is the server where the producer is deployed
	■ `port_number` is the HTTP Listener port number
	■ `context_root` is the Web application's context root.
	■ `providers` is static text. The text entered here depends on how the producer is deployed.
	For example:
	`http://myHost:7778/myEnterprisePortlets/providers`

serviceId	Optional. Service ID of the producer.
	PDK-Java enables you to deploy multiple producers under a single adapter servlet. Producers are identified by their unique service ID. A service ID is required only if the service ID is not appended to the URL end point.
	For example the following URL endpoint requires `sample` as the service ID:
	`http://domain.us.oracle.com:7778/axyz/providers`
	However, the following URL endpoint, does not require a service ID:
	`http://domain.us.oracle.com:7778/axyz/providers/sample`
	The service ID is used to look up a file called `<service_id>.properties`, which defines the characteristics of the producer, such as whether to display its test page. Use any value to create the service ID.

proxyHost	Optional. Host name or IP address of the proxy server. A proxy is required if the WebCenter application and the remote portlet producer are separated by a firewall and an HTTP proxy is needed for communication with the producer.
proxyPort	Optional. Port number on which the proxy server listens.
subscriberID	Optional. Consumer's identifier, if required. When a producer is registered with an application, a call is made to the producer. During the call, the consumer (WebCenter application in this instance) passes the value for Subscriber ID to the producer. If the producer does not see the expected value for Subscriber ID, it might reject the registration call.
sharedKey	Optional. The shared key is used for message authentication with the remote producer. Message authentication ensures that the incoming messages are sent from a host with a shared key. You should enable sessions using the `sharedKey` argument, as well as the `establishSession` argument.
timeout	Optional. Timeout setting for communications with the producer, in seconds. For example, the maximum time the producer may take to register, deregister, or display portlets on WebCenter pages. Individual portlets may define their own timeout period, which takes precedence over the value expressed here.

Argument	Definition
establishSession	Optional. Enable a user session when executing portlets from this producer. Valid values are `true` and `false`. The default for this argument is `false`. You should enable sessions using the `establishSession` argument, as well as the `sharedKey` argument. When sessions are enabled (`true`), the server maintains session-specific information, such as the user name. Message authentication uses sessions, so if a shared key is specified, this option should also be enabled. For sessionless communication between the producer and the server, set to `false`.
externalApp	Optional. Name of the external application associated with this producer.
mapUser	Optional. Flag indicating whether the mapped username from the external application should be passed to the producer. Valid values are `true` and `false`. This argument defaults to `true`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.9.3 Example

The following example changes a PDK-Java producer registered with `MyApp` to use a proxy server.

```
wls:/weblogic/serverConfig> setPDKJavaProducer(appName='MyApp',
name='MyProducer', url='http://myhost.com/jpdk/providers/sample',
proxyHost='myproxy.com', proxyPort=80)
```

5.8.10 deregisterPDKJavaProducer

Module: Oracle WebCenter

Use with WLST: Online

5.8.10.1 Description

Deregisters an Oracle PDK-Java producer and deletes the associated connection, for a named WebCenter application.

5.8.10.2 Syntax

```
deregisterPDKJavaProducer(appName, name, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing PDK-Java producer.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.

Argument	Definition
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.10.3 Example

The following example deregisters the `wc-WebClipping` producer, and deletes the associated connection.

```
wls:/weblogic/serverConfig> deregisterPDKJavaProducer(appName='webcenter',
name='wc-WebClipping')
Already in Domain Runtime Tree
Producer wc-WebClipping has been deregistered.
Already in Domain Runtime Tree
"wc-WebClipping" successfully deleted
Already in Domain Runtime Tree
"wc-WebClipping-urlconn" successfully deleted
```

5.8.11 listPDKJavaProducers

Module: Oracle WebCenter

Use with WLST: Online

5.8.11.1 Description

Lists details for one or more Oracle PDK-Java producers registered with a named WebCenter application.

5.8.11.2 Syntax

```
listPDKJavaProducers(appName, [name],[verbose], [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Optional. Name of an existing PDK-Java portlet producer. If omitted, connection details for all PDK-Java producers configured for this WebCenter application are listed.
verbose	Optional. Displays PDK-Java producer connection details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listPDKJavaProducers` lists all connection properties. When set to `false`, `listPDKJavaProducers` lists connection names only. This argument defaults to `true`. If you set this argument to `false`, do not specify the `name` argument.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.11.3 Example

The following example lists all the connection properties (verbose mode) for the JPDKSamples producer.

```
wls:/weblogic/serverConfig> listPDKJavaProducers(appName='webcenter',
name='JPDKSamples', verbose=true)
----------------------
wc-WebClipping
----------------------
Service Id: None
Shared Key: None
External Application Id: None
Subscriber Id: None
URL: http://myhost.com:9999/portalTools/webClipping/providers/webClipping
----------------------
wc-OmniPortlet
----------------------
Service Id: None
Shared Key: None
External Application Id: None
Subscriber Id: None
URL: http://myhost:9999/portalTools/omniPortlet/providers/omniPortlet
```

5.8.12 refreshProducer

Module: Oracle WebCenter

Use with WLST: Online

5.8.12.1 Description

Refreshes the metadata stored for a named producer to reflect the portlets that are currently offered by that producer.

5.8.12.2 Syntax

```
refreshProducer(appName, producerName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which the producer is registered.
producerName	Name of an existing producer.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.12.3 Example

The following example refreshes the `WSRPSamples` producer in an application named `webcenter`.

```
wls:/weblogic/serverConfig> refreshProducer(appName='webcenter',
ProducerName='WSRPSamples')
Producer WSRPSamples has been refreshed.
```

5.8.13 registerOOTBProducers

Module: Oracle WebCenter

Use with WLST: Online

5.8.13.1 Description

Registers out-of-the-box producers provided with Oracle WebCenter: OmniPortlet, Web Clipping, Rich Text Portlet, and WSRP Tools.

5.8.13.2 Syntax

```
registerOOTBProducers(producerHost, producerPort, appName, [server,
applicationVersion])
```

Argument	Definition
producerHost	Host name or IP address of the server hosting out-of-the-box producers.
producerPort	Port number for the server hosting out-of-the-box producers.
appName	Name of the WebCenter application in which the out-of-the-box producers are to be registered.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.13.3 Example

The following example registers out-of-the-box producers in a WebCenter application named `myApp`.

```
wls:/weblogic/serverConfig> registerOOTBProducers(producerHost='myhost.com',
producerPort=9999, appName='myApp')
Registering Out-of-the-Box Producers
Registering producers at http://myhost.com:9999

Registering Omniportlet
Created connection wc-OmniPortlet-urlconn
Created connection wc-OmniPortlet
Producer connection wc-OmniPortlet has been registered.

Registering WebClipping
Created connection wc-WebClipping-urlconn
Created connection wc-WebClipping
Producer connection wc-WebClipping has been registered.

Registering RichTextPortlet
Created connection wc-RichText-wsconn
Created connection wc-RichText
Producer connection wc-RichText has been registered.

Registering WSRP Tools
Created connection wc-WSRPTools-wsconn
Created connection wc-WSRPTools
Producer connection wc-WSRPTools has been registered.
```

5.8.14 deregisterOOTBProducers

Module: Oracle WebCenter

Use with WLST: Online

5.8.14.1 Description

Deregisters out-of-the-box producers provided with Oracle WebCenter: OmniPortlet, Web Clipping, Rich Text Portlet, and WSRP Tools.

5.8.14.2 Syntax

```
deregisterOOTBProducers(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which the out-of-the-box producers are currently registered.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.14.3 Example

The following example deregisters out-of-the-box WebCenter producers, and deletes their associated connections, in an application named `myApp`.

```
wls:/weblogic/serverConfig> deregisterOOTBProducers(appName='myApp')
Deregistering Out-of-the-Box Producers

Deregistering Omniportlet
Producer wc-OmniPortlet has been deregistered.
wc-OmniPortlet successfully deleted
wc-OmniPortlet-urlconn successfully deleted

Deregistering WebClipping
Producer wc-WebClipping has been deregistered.
wc-WebClipping successfully deleted
wc-WebClipping-urlconn successfully deleted

Deregistering RichTextPortlet
Producer wc-RichText has been deregistered.
wc-RichText successfully deleted
wc-RichText-wsconn successfully deleted

Deregistering WSRP Tools
Producer wc-WSRPTools has been deregistered.
wc-WSRPTools successfully deleted
wc-WSRPTools-wsconn successfully deleted
```

5.8.15 registerSampleProducers

Module: Oracle WebCenter

Use with WLST: Online

5.8.15.1 Description

Registers the sample producers provided with Oracle WebCenter with a named WebCenter application. There are two sample producers — WSRP Samples and JPDK Samples.

5.8.15.2 Syntax

```
registerSampleProducers(producerHost, producerPort, appName, [server,
applicationVersion])
```

Argument	Definition
producerHost	Host name or IP address of the server hosting the sample producers.
producerPort	Port number for the server hosting the sample producers.
appName	Name of the WebCenter application in which the sample producers are to be registered.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.15.3 Example

The following example registers Oracle WebCenter sample producers in an application named `myApp`.

```
wls:/weblogic/serverConfig> registerSampleProducers(producerHost='myhost.com',
producerPort=9999, appName='myApp')
```

5.8.16 deregisterSampleProducers

Module: Oracle WebCenter

Use with WLST: Online

5.8.16.1 Description

Deregisters the Oracle WebCenter sample producers (`WSRP Samples` and `JPDK Samples`) from a named WebCenter application.

5.8.16.2 Syntax

```
deregisterSampleProducers(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which the sample producers are currently registered. If a value is not specified, this argument defaults to `webcenter`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.8.16.3 Example

The following example deregisters sample producers from a WebCenter application named myApp.

```
wls:/weblogic/serverConfig> deregisterSampleProducers(appName='myApp')
```

5.9 RSS

Use the commands listed in Table 5–16 to manage proxy settings for the RSS service in WebCenter Spaces.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 5–16 RSS WLST Commands

Use this command...	To...	Use with WLST...
getRssProxyConfig	Return the proxy host and proxy port used by the RSS service.	Online
setRssProxyConfig	Specify the proxy host and proxy port used by the RSS service.	Online

5.9.1 getRssProxyConfig

Module: Oracle WebCenter

Use with WLST: Online

5.9.1.1 Description

(WebCenter Spaces applications only.) Returns the proxy host and proxy port used by the RSS service. Depending on your network configuration, proxy details may be required to display external RSS news feeds in WebCenter Spaces.

5.9.1.2 Syntax

```
getRssProxyConfig(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application—always webcenter.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.9.1.3 Example

The following example returns the proxy host and proxy port used by the RSS service in a WebCenter Spaces application.

```
wls:/weblogic/serverConfig> getRssProxyConfig(appName='webcenter')
```

5.9.2 setRssProxyConfig

Module: Oracle WebCenter

Use with WLST: Online

5.9.2.1 Description

(WebCenter Spaces applications only.) Specifies the proxy host and port for the RSS service. Depending on your network configuration, proxy details may be required to display external RSS news feeds in WebCenter Spaces.

5.9.2.2 Syntax

```
setRssProxyConfig(appName, proxyHost, proxyPort, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation—always `webcenter`.
proxyHost	Host name of the proxy server.
proxyPort	Port on which the proxy server is running.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.9.2.3 Example

The following example sets the proxy host and proxy port used by the RSS service in a WebCenter Spaces application.

```
wls:/weblogic/serverConfig> setRssProxyConfig(appName='webcenter',
proxyHost='www-proxy.example.com', proxyPort='80')
```

5.10 Search

Use the commands listed in Table 5–17 to manage Oracle Secure Enterprise Search (SES) connections for WebCenter applications.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 5–17 Search WLST Commands

Use this command...	To...	Use with WLST...
createSESConnection	Create a connection to an SES instance for a WebCenter application.	Online
setSESConnection	Edit an existing SES search connection.	Online
listSESConnections	List individual or all SES search connections that are configured for a specific WebCenter application.	Online

Table 5–17 (Cont.) Search WLST Commands

Use this command...	To...	Use with WLST...
setSearchConfig	Modify search settings for a WebCenter application.	Online
setSearchSESConfig	Configure search settings for an existing SES search connection.	Online
listSearchConfig	List search properties for a WebCenter application.	Online
listSearchSESConfig	List SES properties for a WebCenter application.	Online

5.10.1 createSESConnection

Module: Oracle WebCenter

Use with WLST: Online

5.10.1.1 Description

Creates a connection to an Oracle Secure Enterprise Search (SES) instance for a WebCenter application.

5.10.1.2 Syntax

```
createSESConnection(appName, name, url, appUser, appPassword,[default],
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.
url	Web Services URL that Oracle Secure Enterprise Search exposes to enable Search requests. Use the format: `http://<host>:<port>/search/query/OracleSearch`
appUser	User name that the WebCenter application uses to authenticate itself as a trusted application to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter users. The specified user must be present in both the Oracle Identity Management server configured for the WebCenter application and the Oracle Identity Management server configured for Oracle SES.
appPassword	Password for the user name specified.
default	Optional. Configures WebCenter Search service to actively use the search connection. Valid options are `true` and `false`. Setting to `true` replaces any other search connection that is being used. Setting to `false` does not change the current Search service configuration. This argument defaults to `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.10.1.3 Example

The following example creates a new search connection that points to the SES instance `http://myhost.com:7777/search/query/OracleSearch` and makes this connection the active search connection for the WebCenter application named `app1`.

```
wls:/weblogic/serverConfig> createSESConnection(appName='app1', name='SESConn1',
url='http://myhost.com:7777/search/query/OracleSearch', appUser='wpadmin',
appPassword='password', default=true)
```

5.10.2 setSESConnection

Module: Oracle WebCenter

Use with WLST: Online

5.10.2.1 Description

Edits an existing Oracle Secure Enterprise Search (SES) search connection.

5.10.2.2 Syntax

```
setSESConnection(appName, name, [url], [appUser],[appPassword],[default],
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing search connection.
url	Optional. Web Services URL that Oracle Secure Enterprise Search exposes to enable Search requests. Use the format: `http://<host>:<port>/search/query/OracleSearch`
appUser	Optional. User name that the WebCenter application uses to log in to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter users.
appPassword	Optional. Password that the WebCenter application uses to log in to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter users.
default	Optional. Configures WebCenter Search service to actively use the search connection. Valid options are `true` and `false`. Setting to `true` replaces any other search connection that is being used. Setting to `false` does not change the current Search service configuration. This argument defaults to `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.10.2.3 Example

The following example modifies the URL of a search connection named `SESConn1` and makes the connection the active search connection for a WebCenter application named `app1`.

```
wls:/weblogic/serverConfig> setSESConnection(appName='app1', name='SESConn1',
url='http://myhost.com:7777/search/query/OracleSearch', appUser='wpadmin',
appPassword='password', default=true)
```

5.10.3 listSESConnections

Module: Oracle WebCenter

Use with WLST: Online

5.10.3.1 Description

Lists the names of all Oracle Secure Enterprise Search (SES) search connections configured for a WebCenter application.

5.10.3.2 Syntax

```
listSESConnections(appName, [verbose], [name], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application for which you want to perform this operation.
verbose	Optional. Displays search connection details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listSESConnections` lists all of the search connections that are configured for a WebCenter application, along with their details. When set to `false`, `listSESConnections` lists connection names only. This argument defaults to `false`. If you set this argument to `false`, do not specify the `name` argument.
name	Optional. Name of an existing search connection. You can use this argument to view details about a specific connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.10.3.3 Examples

The following example displays connection details for all search connections configured for a WebCenter application, named `WebCenterApp`.

```
wls:/weblogic/serverConfig> listSESConnections(appName='WebCenterApp',
verbose='true')
```

The following example displays connection details for a search connection named `SESConn1`.

```
wls:/weblogic/serverConfig> listSESConnections(appName='WebCenterApp',
verbose='true', name='SESConn1')
```

5.10.4 setSearchConfig

Module: Oracle WebCenter

Use with WLST: Online

5.10.4.1 Description

Modifies search settings for a WebCenter application. If a parameter is not specified it is not modified.

5.10.4.2 Syntax

```
setSearchConfig(appName,[numSavedSearches],[numResultsRegion],[numResultsMain],
[numResultsToolbar],[executionTimeout],[prepareTimeout],[showAllExecutionTimeout],
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
numSavedSearches	Optional. The number of saved searches to display in the Saved Searches drop down (on main search page).
numResultsRegion	Optional. The number of saved searches displayed in a Saved Search task flow.
numResultsMain	Optional. The number of search results displayed, per service, for searches submitted from the main search page.
numResultsToolbar	Optional. The number of search results displayed, per service, for searches submitted from the global search toolbar. The value for this argument must be a valid number.
executionTimeout	Optional. The maximum time that a service is allowed to execute a search (in ms). The value for this argument must be a valid number.
prepareTimeout	Optional. The maximum time that a service is allowed to initialize a search (in ms). The value for this argument must be a valid number.
showAllExecutionTimeout	Optional. The maximum time that a service is allowed to display search all results (in ms). The value for this argument must be a valid number.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.10.4.3 Examples

The following example specifies that saved searches display five search results per service. Additionally, that a seven second search execution timeout is required.

```
wls:/weblogic/serverConfig> setSearchConfig(appName='webcenter',
numResultsRegion=5, executionTimeout=7000);
```

The following example increases the number of saved searches in the Saved Searches drop down list to eight, while changing the number of results from global toolbar searches to ten, per service.

```
wls:/weblogic/serverConfig> setSearchConfig(appName='webcenter',
numSavedSearches=8, numResultsToolbar=10);
```

The following example sets the search execution timeout to five seconds and allows each service fifteen seconds to display search results before timing out.

```
wls:/weblogic/serverConfig> setSearchConfig(appName='webcenter',
```

```
executionTimeout=5000, showAllExecutionTimeout=15000);
```

5.10.5 setSearchSESConfig

Module: Oracle WebCenter

Use with WLST: Online

5.10.5.1 Description

Configures search settings for an existing Oracle Secure Enterprise Search (SES) search connection. If a parameter is not specified it is not modified.

5.10.5.2 Syntax

```
setSearchSESConfig(appName,[connectionName],[dataGroup],[topNRows], [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
connectionName	Optional. Names the search connection that the Search service must use.
dataGroup	Optional. Names the Secure Enterprise Search data group in which to search. If a value is not provided, everything in the Oracle Secure Enterprise Search instance will be searched.
topNRows	Optional. Number of top N rows of search results for gathering refinement data.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.10.5.3 Example

The following example specifies that the Search service must use the search connection named `SESConn1`, and to search the data group named `group2`.

```
wls:/weblogic/serverConfig> setSearchSESConfig
(appName='webcenter',connectionName='SESConn1', dataGroup='group2',
topNRows=200);
```

The following example changes the maximum number of search results that the Search service returns. No connection name is specified, in this example, so this configuration change is applied to the current default (or active) search connection.

```
wls:/weblogic/serverConfig> setSearchSESConfig(appName='webcenter', topNRows=500);
Already in Domain Runtime Tree
Restart is needed for the service connection changes to take effect.
```

5.10.6 listSearchConfig

Module: Oracle WebCenter

Use with WLST: Online

5.10.6.1 Description

Lists search settings for a WebCenter application.

5.10.6.2 Syntax

```
listSearchConfig(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application for which you want to perform this operation.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.10.6.3 Example

The following example displays search configuration information for a WebCenter application named `webcenter`.

```
wls:/weblogic/serverConfig> listSearchConfig(appName='webcenter')
```

5.10.7 listSearchSESConfig

Module: Oracle WebCenter

Use with WLST: Online

5.10.7.1 Description

Lists SES search settings for a WebCenter application.

5.10.7.2 Syntax

```
listSearchSESConfig(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application for which you want to perform this operation.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.10.7.3 Example

The following example displays SES search configuration information for a WebCenter application named `webcenter`.

```
wls:/weblogic/serverConfig> listSearchSESConfig(appName='webcenter')
Already in Domain Runtime Tree
-----------------
Search SES Config
```

```
-----------------
connectionName:  SESConn1
dataGroup:  group2
topNRows:  200
```

5.11 Worklists

Use the commands listed in Table 5–18 to manage BPEL server connections for WebCenter applications.

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is deployed. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 5–18 Worklist Commands

Use this command...	To...	Use with WLST...
createBPELConnection	Create a connection to a BPEL server for a WebCenter application.	Online
setBPELConnection	Edit an existing BPEL server connection.	Online
listBPELConnections	List all of the BPEL server connections that are configured for a WebCenter application.	Online
addWorklistConnection	Enable an existing BPEL server connection for the Worklist service.	Online
removeWorklistConnection	Disable a BPEL server connection currently used by the Worklist service.	Online
listWorklistConnections	List individual or all BPEL server connections configured for the Worklist service.	Online

5.11.1 createBPELConnection

Module: Oracle WebCenter

Use with WLST: Online

5.11.1.1 Description

Creates a connection to a BPEL server for a named WebCenter application. BPEL server connections can be used by the application's Worklist service and WebCenter Spaces workflows.

To configure the Worklist service to actively use a new BPEL server connection, use the `addWorklistConnection` command. See Section 5.11.4, "addWorklistConnection".

To specify the BPEL server connection that WebCenter Spaces uses for its internal workflows, use the `setSpacesWorkflowConnectionName` command. See Section 5.12.2, "setSpacesWorkflowConnectionName".

5.11.1.2 Syntax

```
createBPELConnection(appName, name, url, [policy], [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter application.
url	URL required to access the BPEL server. Use the format: `<protocol>://<host>:<port>` The BPEL server URL must be unique within the WebCenter application.
policy	Optional. SAML token policy this connection uses for authentication. This argument defaults to `oracle/wss10_saml_token_client_policy`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.11.1.3 Examples

The following example creates a connection named `WebCenter Worklist` with the URL value provided:

```
wls:/weblogic/serverConfig> createBPELConnection(appName='webcenter',
name='WebCenter Worklist', url='http://myhost.com:8001',
policy='oracle/wss10_saml_token_client_policy)
```

The following example creates a connection that uses the default security policy:

```
wls:/weblogic/serverConfig> createBPELConnection(appName='webcenter',
name='WebCenter Worklist', url='http://myhost.com:8001')
```

5.11.2 setBPELConnection

Module: Oracle WebCenter

Use with WLST: Online

5.11.2.1 Description

Edits an existing BPEL server connection. The BPEL server URL and the security policy are both editable.

To configure the Worklist service to actively use an existing BPEL server connection, use the `addWorklistConnection` command. See Section 5.11.4, "addWorklistConnection".

To specify the BPEL server connection used for Webcenter Spaces workflows, use the `setSpacesWorkflowConnectionName` command. See Section 5.12.2, "setSpacesWorkflowConnectionName".

5.11.2.2 Syntax

```
setBPELConnection(appName, name,[url],[policy],[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Existing BPEL server connection name.
url	Optional. URL required to access the BPEL server. Use the format: <protocol>://<host>:<port> The BPEL server URL must be unique within the WebCenter application.
policy	Optional. SAML token policy this connection uses for authentication. This argument defaults to oracle/wss10_saml_token_client_policy.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, WLS_Spaces. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.11.2.3 Examples

The following example updates the BPEL server URL and security policy for a connection named WebCenter Worklist.

```
wls:/weblogic/serverConfig> setBPELConnection(appName='webcenter',
name='WebCenter Worklist',url='http://myhost.com:6666', policy='oracle/wss10_
saml_token_with_message_protection_client_policy')
```

The following example updates the BPEL server URL for a connection named WebCenter Worklist.

```
wls:/weblogic/serverConfig> setBPELConnection(appName='webcenter',
name='WebCenter Worklist',url='http://myhost.com:8001')
```

5.11.3 listBPELConnections

Module: Oracle WebCenter

Use with WLST: Online

5.11.3.1 Description

Without any arguments, this command lists all the BPEL connections that are configured for a specific WebCenter application. All BPEL connections are listed, even connections not currently used.

5.11.3.2 Syntax

```
listBPELConnections(appName, [verbose], [name], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application for which you want to list BPEL server connections.

Argument	Definition
verbose	Optional. Displays BPEL server connection details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listBPELConnections` lists all of the BPEL server connections that are configured, along with their details. When set to `false`, `listBPELConnections` lists connection names only. This argument defaults to `false`. If you set this argument to `false`, do not specify the `name` argument.
name	Optional. Name of an existing BPEL server connection. You can use this argument to view details about a specific connection. To list all the connections, omit the `name` argument.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.11.3.3 Examples

The following example lists the names of all the BPEL server connections that are configured for a WebCenter application.

```
wls:/weblogic/serverConfig> listBPELConnections(appName='webcenter')
  ------------------
  WebCenter Worklist
  ------------------
  ------------------
  Human Resources Worklist
  ------------------
```

The following example lists the names and details of all of the BPEL server connections that are configured for a WebCenter application.

```
wls:/weblogic/serverConfig> listBPELConnections(appName='webcenter', verbose=true)
  ------------------
  WebCenter Worklist
  ------------------
  Connection Name: WebCenter Worklist
  PolicyURI:oracle/wss10_saml_token_client_policy
  URL:http://myhost.com:8001
  ------------------
  Human Resources Worklist
  ------------------
  Connection Name: Human Resources Worklist
  PolicyURI:oracle/wss10_saml_token_client_policy
  URL:http://myhost.com:8888
  ------------------
```

5.11.4 addWorklistConnection

Module: Oracle WebCenter

Use with WLST: Online

5.11.4.1 Description

Enable an existing BPEL server connection for Worklist services. The Worklist service supports multiple connections so that WebCenter users can monitor and manage assignments and notifications from a range of BPEL servers.

The name must specify an existing BPEL server connection.

5.11.4.2 Syntax

```
addWorklistConnection(appName, name,[verbose, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing BPEL server connection.
verbose	Optional. Displays output indicating whether a matching BPEL server connection exists and provides connection details. `True` turns verbose mode on; `false` turns verbose mode off. This argument defaults to `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.11.4.3 Examples

The following example enables the `Human Resources Worklist` connection for the Worklist service.

```
wls:/weblogic/serverConfig> addWorklistConnection(appName='webcenter',
name='Human Resources Worklist', verbose=true)
  Human Resources Worklist successfully added to WorkList
  ------------------
  Human Resources Worklist
  ------------------
  Connection Name: Human Resources Worklist
  PolicyURI:oracle/wss10_saml_token_client_policy
  URL:http://myhost.com:8888
```

The following example also enables the `Human Resources Worklist` connection for the Worklist service.

```
wls:/weblogic/serverConfig> addWorklistConnection(appName='webcenter',
name='Human Resources Worklist', verbose=true)
  Human Resources Worklist successfully added to WorkList
  ------------------
  Human Resources Worklist
  ------------------
  Connection Name: Human Resources Worklist
  PolicyURI:oracle/oracle/wss10_saml_token_client_policy
  URL:http://myhost.com:8888
```

5.11.5 removeWorklistConnection

Module: Oracle WebCenter

Use with WLST: Online

5.11.5.1 Description

Disables a BPEL server connection that is currently used by the Worklist service. Connection details are retained but the Worklist service no longer uses the connection specified.

5.11.5.2 Syntax

```
removeWorklistConnection(appName, name, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
name	Name of an existing BPEL server connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.11.5.3 Example

The following example disables the BPEL server connection named `WebCenter Worklist` for the Worklist service.

```
wls:/weblogic/serverConfig> removeWorklistConnection(appName='webcenter',
name='WebCenter Worklist')
  WebCenter Worklist successfully removed from WorkList
```

5.11.6 listWorklistConnections

Module: Oracle WebCenter

Use with WLST: Online

5.11.6.1 Description

Without any arguments, this command lists all of the BPEL server connections that are configured for the Worklist service, in a named WebCenter application.

5.11.6.2 Syntax

```
listWorklistConnections(appName, [verbose],[name], [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application for which you want to perform this operation.

Argument	Definition
verbose	Optional. Displays BPEL server connection details in verbose mode. Valid options are `true` and `false`. When set to `true`, `listWorklistConnections` lists all of the BPEL server connections that are configured for the Worklist service, along with their details. When set to `false`, `listWorklistConnections` lists connection names only. This argument defaults to `false`. If you set this argument to `false`, do not specify the `name` argument.
name	Optional. Name of an existing BPEL server connection. You can use this argument to view details about a specific connection. To list all connections, omit the `name` argument.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.11.6.3 Examples

The following example lists the names of all of the BPEL server connections that are configured for the Worklist service.

```
wls:/weblogic/serverConfig> listWorklistConnections(appName='webcenter')
  ------------------
  WebCenter Worklist
  ------------------
```

The following example lists both the names and connection details of all of the BPEL server connections that are configured for the Worklist service.

```
wls:/weblogic/serverConfig> listWorklistConnections(appName='webcenter',
verbose=true)
  ------------------
  WebCenter Worklist
  ------------------
  Connection Name: WebCenter Worklist
  PolicyURI:oracle/wss10_saml_token_client_policy
  URL:http://myhost.com:8001
```

The following example lists connection details of a named BPEL server connection—`MyWorklist`. As the Worklist service is not currently configured to use `MyWorklist`, an appropriate message displays.

```
wls:/weblogic/serverConfig> listWorklistConnections(appName='webcenter',
verbose=true, name='MyWorklist')
  ------------------
  The following connection is not in the ADF Worklist:MyWorklist
```

5.12 WebCenter Spaces Workflows

Use the commands listed in Table 5–19 to manage the BPEL server connection used for WebCenter Spaces workflows (group space membership notifications, group space subscription requests, and so on).

Configuration changes made using these WebCenter WLST commands are only effective after your restart the Managed Server on which the WebCenter application is

deployed. For details, see *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 5–19 WebCenter Spaces WLST Commands

Use This Command...	To...	Use with WLST...
getSpacesWorkflowConnectionName	Return the name of the BPEL server connection that WebCenter Spaces is using for internal workflows.	Online
setSpacesWorkflowConnectionName	Specify the BPEL server connection used for Webcenter Spaces workflows.	Online

5.12.1 getSpacesWorkflowConnectionName

Module: Oracle WebCenter

Use with WLST: Online

5.12.1.1 Description

Returns the name of the BPEL server connection that WebCenter Spaces is currently using for internal workflows (group space membership notifications, group space subscription requests, and so on).

5.12.1.2 Syntax

```
getSpacesWorkflowConnectionName(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application—always `webcenter`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.12.1.3 Example

The following example names the BPEL server connection that WebCenter Spaces is currently using for internal workflow.

```
wls:/weblogic/serverConfig> getSpacesWorkflowConnectionName(appName='webcenter')
WorkflowConfigConnectionName: WebCenter-Worklist
```

5.12.2 setSpacesWorkflowConnectionName

Module: Oracle WebCenter

Use with WLST: Online

5.12.2.1 Description

Specifies the BPEL server connection that Webcenter Spaces uses for internal workflows. WebCenter Spaces uses a BPEL server included with the Oracle SOA Suite to host internal workflows, such as group space membership notifications, group space subscription requests, and so on. The connection name specified here must be a valid BPEL server connection.

5.12.2.2 Syntax

```
setSpacesWorkflowConnectionName(appName, name, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application—always `webcenter`.
name	Name of an existing BPEL connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.12.2.3 Example

The following example specifies that WebCenter Spaces uses the BPEL server connection named `WebCenter-Worklist` for its internal workflows.

```
wls:/weblogic/serverConfig>setSpacesWorkflowConnectionName(appName='webcenter',
name='WebCenter-Worklist')
```

5.13 Import and Export

Use the commands listed in Table 5–20 to export and import WebCenter applications and producer metadata.

Table 5–20 Import and Export WLST Commands

Use this command...	To...	Use with WLST...
exportWebCenterApplication	Export WebCenter Spaces to a WebCenter export archive.	Online
importWebCenterApplication	Import WebCenter Spaces from a WebCenter export archive.	Online
exportGroupSpaces	Export one or more group spaces to a WebCenter export archive.	Online
exportGroupSpaceTemplates	Export one or more group space templates to a WebCenter export archive.	Online
importGroupSpaces	Import one or more group spaces or group space templates from a WebCenter export archive.	Online
exportProducerMetadata	Export portlet client metadata to a WebCenter export archive.	Online
importProducerMetadata	Import portlet client metadata from a WebCenter export archive.	Online

5.13.1 exportWebCenterApplication

Module: Oracle WebCenter

Use with WLST: Online

5.13.1.1 Description

(WebCenter Spaces only) Exports a WebCenter Spaces application to a WebCenter export archive (.EAR) using the filename provided.

5.13.1.2 Syntax

```
exportWebCenterApplication(appName, fileName, [exportCustomizations,
exportSecurity, exportData, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application in which you want to perform this operation.
fileName	Name of the export archive EAR file to which you want the export to be written.
exportCustomizations	Optional. Valid values are `true` and `false`. When set to `true`, all customizations are exported. When set to `false`, customizations are not exported, that is, default task flows are exported without any customizations. This argument defaults to `true`.
exportSecurity	Optional. Valid values are `true` and `false`. When set to `true`, policy-store.xml contains application roles and permissions, as well as user details and their role assignments. When set to `false`, policy-store.xml contains application roles and permissions only. User details are not exported. This argument defaults to `false`.
exportData	Optional. Valid values are `true` and `false`. When set to `true`, data stored in the WebCenter Spaces database for lists, events, tags, and links is exported. When set to `false`, this data is not exported. This argument defaults to `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.13.1.3 Examples

The following example exports a WebCenter Spaces application and all possible data to a file named `myExport.ear`.

```
wls:/weblogic/serverConfig> exportWebCenterApplication(appName='webcenter',
fileName='myExport.ear', exportCustomizations=true, exportSecurity=true,
exportData=true)
```

The following example exports a test application with the intention of importing the resultant EAR to an alternative system with a different user base. In this case, security policies (which might reference users or roles specific to the originating server) are not required. Additionally, data created during testing (such as lists, group space events, links, tags) is not required.

```
wls:/weblogic/serverConfig> exportWebCenterApplication(appName='webcenter',
fileName='export.ear')
```

5.13.2 importWebCenterApplication

Module: Oracle WebCenter

Use with WLST: Online

5.13.2.1 Description

(WebCenter Spaces only) Imports a WebCenter Spaces application from a WebCenter export archive file to a server.

After importing WebCenter Spaces you will need to restart the managed server where the application is deployed.

5.13.2.2 Syntax

```
importWebCenterApplication(appName, fileName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application in which you want to perform this operation.
fileName	Name of the WebCenter export archive that you want to import.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.13.2.3 Example

The following example imports WebCenter Spaces from the export archive `myExport.ear`.

```
wls:/weblogic/serverConfig> importWebCenterApplication(appName='webcenter',
fileName='myExport.ear')
```

5.13.3 exportGroupSpaces

Module: Oracle WebCenter

Use with WLST: Online

5.13.3.1 Description

(WebCenter Spaces only) Exports one or more group spaces to a WebCenter export archive (.EAR), using the filename specified.

5.13.3.2 Syntax

```
exportGroupSpaces(appName, fileName, names, string, [exportCustomizations,
exportSecurity, exportData, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application in which you want to perform this operation—always `webcenter`.
fileName	Name of the local file to which the export will be written.
names	Names of the group spaces that you want to export. Separate multiple group space names with a comma.

Argument	Definition
exportCustomizations	Optional. Valid values are `true` and `false`. When set to `false`, customizations are not exported, that is, default task flows are exported without any customizations. This argument defaults to `true`.
exportSecurity	Optional. Valid values are `true` and `false`. When set to `true`, policy-store.xml contains group space roles and permissions, as well as member details and their role assignments. When set to `false`, policy-store.xml contains group space roles and permissions only. Member details are not exported. This argument defaults to `false`.
exportData	Optional. Valid values are `true` and `false`. When set to `true`, group space data stored in the WebCenter Spaces database for lists, events, tags, and links is exported. When set to `false`, this data is not exported. This argument defaults to `false`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.13.3.3 Example

The following example exports two group spaces (`myGroup1` and `myGroup2`) from a WebCenter Spaces application named `webcenter`.

```
wls:/weblogic/serverConfig> exportGroupSpaces(appName='webcenter',
fileName='myExport.ear', names='myGroup1, myGroup2', exportCustomizations=true,
exportSecurity=true, exportData=true)
```

5.13.4 exportGroupSpaceTemplates

Module: Oracle WebCenter

Use with WLST: Online

5.13.4.1 Description

(WebCenter Spaces only) Exports one or more group space templates to a WebCenter export archive (.EAR), using the filename specified.

5.13.4.2 Syntax

```
exportGroupSpaceTemplates(appName, fileName, names, string, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application in which you want to perform this operation—always `webcenter`.
fileName	Name of the local file to which the export will be written.
groupSpaceTempNames	Names of the group space templates that you want to export. Separate multiple group space template names with a comma.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.13.4.3 Example

The following example exports two group space templates (`myGroupTemplate1` and `myGroupTemplate2`) from a WebCenter Spaces application named `webcenter`.

```
wls:/weblogic/serverConfig> exportGroupSpaceTemplates(appName='webcenter',
fileName='myExport.ear', names='myGroupTemplate1, myGroupTemplate2')
```

5.13.5 importGroupSpaces

Module: Oracle WebCenter

Use with WLST: Online

5.13.5.1 Description

(WebCenter Spaces only) Imports one or more group spaces or group space templates from a WebCenter export archive.

5.13.5.2 Syntax

```
importGroupSpaces(appName, fileName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Spaces application in which you want to perform this operation—always `webcenter`.
fileName	Name of the WebCenter archive file that you want to import.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.13.5.3 Example

The following example imports group spaces or group space templates from a WebCenter archive named `myExport.ear` to a WebCenter Spaces application named `webcenter`.

```
wls:/weblogic/serverConfig> importGroupSpaces(appName='webcenter',
fileName='myExport.ear')
```

5.13.6 exportProducerMetadata

Module: Oracle WebCenter

Use with WLST: Online

5.13.6.1 Description

Exports portlet client metadata to a WebCenter export archive (.EAR), using the filename specified.

This command is not applicable for WebCenter Spaces applications.

5.13.6.2 Syntax

```
exportProducerMetadata(appName, fileName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
fileName	Name of the export archive (.EAR) to which you want the export to be written.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.13.6.3 Example

The following example exports portlet client metadata to an export archive named `myExport.ear`.

```
wls:/weblogic/serverConfig> exportProducerMetadata(appName='myApp',
fileName='myExport.ear')
```

5.13.7 importProducerMetadata

Module: Oracle WebCenter

Use with WLST: Online

5.13.7.1 Description

Imports portlet client metadata from a named WebCenter export archive.

This command is not applicable for WebCenter Spaces applications.

5.13.7.2 Syntax

```
importProducerMetadata(appName, fileName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which you want to perform this operation.
fileName	Name of the WebCenter export archive that you want to import.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WLS_Spaces`. Required when applications with the same name are deployed to different servers and also when you have a cluster.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

5.13.7.3 Example

The following example imports portlet client metadata from a WebCenter export archive named `myExport.ear`.

```
wls:/weblogic/serverConfig> importProducerMetadata(appName='app1',
fileName='myExport.ear')
```

6

User Messaging Service (UMS) Custom WLST Commands

Use the User Messaging Service commands, listed in Table 6–1, to download user messaging preferences from your backend database.

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Table 6–1 User Messaging Service for WLST Configuration

Command category	Description
Section 6.1, "UMS WLST Command Group"	(Brief description of command category)

6.1 UMS WLST Command Group

The UMS WLST commands are listed under the command group "ums".

6.1.1 manageUserMessagingPrefs

Command Category: UMS

Use with WLST: Offline

6.1.1.1 Description

`manageUserMessagingPrefs` is used to download the user messaging preferences from a backend database to the specified xml file, or to upload the user messaging preferences from an XML file into the backend database.

6.1.1.2 Syntax

```
manageUserMessagingPrefs (operation=, filename, url, username, password,
[encoding], [guid], [merge] )
```

Argument	Definition
`operation`	specifies the upload or download operation to be performed.

Argument	Definition
filename	For download, a unique file name (path) to download the user preferences to. For example, /tmp/download.xml (Linux) or C:\temp\download.xml (Windows). For upload, the file name (path) from which to upload the user preferences.
url	The JNDI URL to access the User Messaging Server. For example: t3://<hostname>:<port>
username	The username with login permission to access the User Messaging Server.
password	The password of the username.
encoding	Character encoding to use to download the user preferences.
guid	The globally unique identifier (guid) of a list of users to use to download their preferences. If no guid is specified, the preferences for all users are downloaded.
merge	This option is for upload only. Valid values are: create_new (default): Create new user device, device addresses and/or ruleset entities. An exception will be thrown if an entity with the same primary key already exists and processing will terminate. overwrite: Remove all existing entities of a user and then create new entities. append: Only upload entities that do not already exist.

6.1.1.3 Examples

To download the user messaging preferences of all users to the specified file.

```
wls:offline> manageUserMessagingPrefs(operation='download',
filename='download.xml', url='t3://localhost:8001', username='weblogic',
password='<password>')
```

To download the user messaging preferences of all users to the specified file using UTF-8 character encoding.

```
wls:offline> manageUserMessagingPrefs(operation='download',
filename='download.xml', url='t3://localhost:8001', username='weblogic',
password='<password>', encoding='UTF-8')
```

To download the user messaging preferences of the user with guid 'john.doe' to the specified file.

```
wls:offline> manageUserMessagingPrefs(operation='download',
filename='download.xml', url='t3://localhost:8001', username='weblogic',
password='<password>', guid='john.doe')
```

To download the user messaging preferences of the users with guid 'john.doe' and 'jane.doe' to the specified file using UTF-8 character encoding.

```
wls:offline> manageUserMessagingPrefs(operation='download',
filename='download.xml', url='t3://localhost:8001', username='weblogic',
password='<password>', guid='john.doe,jane.doe', encoding='UTF-8')
```

To upload the user messaging preferences from the specified file to the backend database.

```
wls:offline> manageUserMessagingPrefs(operation='upload', filename='upload.xml',
url='t3://localhost:8001', username='weblogic', password='<password>')
```

To upload the user messaging preferences from the specified file to the backend database and overwrite existing preferences.

```
wls:offline> manageUserMessagingPrefs(operation='upload', filename='upload.xml',
url='t3://localhost:8001', username='weblogic', password='<password>',
merge='overwrite')
```

6.1.2 deployUserMessagingDriver

Command Category: UMS

Use with WLST: Online

6.1.2.1 Description

`deployUserMessagingDriver` is used to deploy additional instances of user messaging drivers.

Specify a base driver type (for example: email, xmpp, voicexml, and others) and a short name for the new driver deployment. The string *usermessagingdriver-* will be prepended to the specified application name. Any valid parameters for the *deploy* command can be specified, and will be passed through when the driver is deployed.

6.1.2.2 Syntax

```
deployUserMessagingDriver(baseDriver, appName, [targets], [stageMode],
[options])
```

Argument	Definition
`baseDriver`	Specifies the base messaging driver type. Must be a known driver type, such as 'email', 'proxy', 'smpp', 'voicexml', or 'xmpp'.
`appName`	A short descriptive name for the new deployment. The specified value will be prepended with the string *usermessagingdriver-*

`targets`
`stageMode`
`options`
	Optional. Additional arguments that are valid for the *deploy* command can be specified and will be passed through when the new driver is deployed.

6.1.2.3 Examples

To deploy a second instance of an email driver with name *myEmail*.

```
wls:base_domain/servereConfig> deployUserMessagingDriver(baseDriver='email',
appName='myEmail')
```

To deploy a second instance of an email driver, specifying deployment targets.

```
wls:base_domain/servereConfig> deployUserMessagingDriver(baseDriver='email',
appName='email2', targets='server1,server2')
```

7

DMS Custom WLST Commands

Use the Dynamic Monitoring Service (DMS) commands to view a specific performance metric, a set of performance metrics, or all performance metrics for a particular server or component.

For additional details about metrics, see the chapter "Monitoring Oracle Fusion Middleware" in the *Oracle Fusion Middleware Administrator's Guide*.

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

7.1 DMS Commands

Use the commands in Table 7–1 to view information about performance metrics.

Table 7–1 DMS Commands

Use this command...	To...	Use with WLST...
displayMetricTableNames	Displays the names of the available DMS metric tables.	Online
displayMetricTables	Displays the content of the DMS metric tables.	Online
dumpMetrics	Displays available metrics.	Online
reloadMetricRules	Reloads the metric rules.	Online

7.1.1 displayMetricTableNames

Use with WLST: Online

7.1.1.1 Description

Displays the names of the available DMS metric tables. The returned value is a string array containing metric table names.

7.1.1.2 Syntax

```
displayMetricTableNames([servers])
```

Argument	Definition

servers	Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic server names and OPMN-managed component names.
	To specify one server, use the following syntax:
	`servers='servername'`
	To specify multiple servers, use one of the following syntax options:
	`servers=['servername1', 'servername2', ...]`
	`servers=('servername1', 'servername2', ...)`
	If this argument is not specified, the command returns the list of metric table names from all WebLogic servers and OPMN-managed components.

7.1.1.3 Example

The following example displays metric table names from all WebLogic servers and OPMN-managed components:

```
displayMetricTableNames()
ADF
ADFc
ADFc_Metadata_Service
ADFc_Region
ADFc_Taskflow
ADFc_Viewport
BAM_common_connectionpool
BAM_common_connectionpool_main
BAM_common_messaging
BAM_common_messaging_consumers
.
.
.
```

The following example displays metric table names for the WebLogic Managed Server called soa_server1:

```
displayMetricTableNames(servers='soa_server1')
ADF
JVM
JVM_ClassLoader
JVM_Compiler
JVM_GC
JVM_Memory
JVM_MemoryPool
JVM_MemorySet
JVM_OS
JVM_Runtime
.
.
.
```

The following example displays metric table names for two WebLogic Managed Servers:

```
displayMetricTableNames(servers=['soa_server1', 'bam-server1'])
ADF
ADFc
ADFc_Metadata_Service
```

```
ADFc_Region
ADFc_Taskflow
ADFc_Viewport
BAM_common_connectionpool
BAM_common_connectionpool_main
BAM_common_messaging
BAM_common_messaging_consumers
.
.
.
```

7.1.2 displayMetricTables

Use with WLST: Online

7.1.2.1 Description

Displays the content of the DMS metric tables.

The returned value is an array of JMX javax.management.openmbean.CompositeData objects. Each array element has the following fields:

- The Table field is the metric table name.
- The Schema field is a javax.management.openmbean.TabularData object containing the metric table schema information.
- The Rows field is a javax.management.openmbean.TabularData object containing the metric table Rows.

The javax.management.openmbean.TabularData object for the metric table schema contains the following four fields:

- The Column field contains the name of the column.
- The Type field contains the type of the column value.
- The Unit field contains the unit of the column.
- The Description field contains the description of the column.

The javax.management.openmbean.TabularData object for the metric rows contains a field for every metric. It uses the metric name as the field name.

7.1.2.2 Syntax

```
displayMetricTables([metricTable_1], [metricTable_2], [...,] [servers]
[variables])
```

Argument	Definition
metricTable_n	Optional. Specifies a list of metric tables. By default, this argument displays all available metrics. The metric table name can contain special characters for simple pattern. matching. The character '?' matches any single character. The character '*' matches zero or more characters. You specify the name of the metric, without an argument name. You can specify zero or more metric table names, in a comma-separated list.

Argument	Definition

servers	Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic server names and OPMN-managed component names.
	To specify one server, use the following syntax:
	`servers='servername'`
	To specify multiple servers, use one of the following syntax options:
	`servers=['servername1', 'servername2', ...]`
	`servers=('servername1', 'servername2', ...)`
	If this argument is not specified, the command returns the list of metric tables from all WebLogic servers and OPMN-managed components.

variables	Optional. Defines the metric aggregation parameters. Valid values are a set of name-value pairs in a Jython map. It uses the following syntax: `variables={name1:value1, name2:value2, ...}` The specific name-value pairs depend on the aggregated metric tables. Each aggregated metric table has its specific set of variable names.

7.1.2.3 Example

The following example displays the data from the JVM and the weblogic.management.runtime.WebAppComponentRuntimeMBean metric tables, and limits it to data retrieved from Server-0 and Server-2:

```
displayMetricTables('JVM','weblogic.management.runtime.WebAppComponentRuntimeMBean',
    servers=['soa_server1','bam_server1'])
.
.
.
ApplicationRuntime:     soa-infra
ComponentName:  /integration/services/IdentityService
ContextRoot:    /integration/services/IdentityService
DeploymentState:        2
FilterDispatchedRequestsEnabled:        false
IndexDirectoryEnabled:  false
JSPDebug:       false
JSPKeepGenerated:       false
JSPPageCheckSecs:       1
JSPVerbose:     true
ModuleId:       /integration/services/IdentityService
ModuleURI:      IdentityService.war
Name:   soa_server1_/integration/services/IdentityService
ObjectName:     com.bea:ApplicationRuntime=soa-infra,Name=soa_server1_
/integration/services/IdentityService,
 ServerRuntime=soa_server1,Type=WebAppComponentRuntime
OpenSessionsCurrentCount:       0
OpenSessionsHighCount:  0
.
.
.
```

The following example displays the aggregated metric tables with the specified metric aggregation parameters:

```
displayMetricTables('j2ee_application:webservices_port_rollup',
      servers=['soa_server1','bam_server1'],
      variables={'host':'hostname', 'servletName':'dms'})
----------------------------------------
j2ee_application:webservices_port_rollup
```

```
----------------------------------------

Faults: 0
Requests:       0
Requests.averageTime:   0.0
Requests.totalTime:     0.0
ServerName:     soa_server1
moduleName:     RuntimeConfigService
moduleType:     WEBs
portName:       RuntimeConfigServicePortSAML
processRequest.active:  0
service.throughput:     0.0
service.time:   0.0
startTime:      1238182359291
webserviceName: RuntimeConfigService

Faults: 0
Requests:       0
Requests.averageTime:   0.0
Requests.totalTime:     0.0
ServerName:     soa_server1
moduleName:     TaskMetadataService
moduleType:     WEBs
portName:       TaskMetadataServicePort
processRequest.active:  0
service.throughput:     0.0
service.time:   0.0
startTime:      1238182358096
webserviceName: TaskMetadataService
.
.
.
```

The following example displays the metric tables which names match the specified patterns:

```
displayMetricTables('J??', 'JVM_*')
.
.
.
---------------
JVM_ThreadStats
---------------

Host:   hostname.us.oracle.com
JVM:    JVM
Name:   threads
Parent: /JVM/MxBeans
Process:        AdminServer:9001
ServerName:     AdminServer
contention.value:       enabled in JVM
daemon.value:   60      threads
deadlock.value: 0       threads
live.value:     61      threads
peak.value:     66      threads
started.value:  241     threads

Host:   hostname.us.oracle.com
JVM:    JVM
Name:   threads
Parent: /JVM/MxBeans
```

```
Process:        soa_server1:9001
ServerName:     soa_server1
contention.value:       enabled in JVM
daemon.value:   68      threads
deadlock.value: 0       threads
live.value:     74      threads
peak.value:     74      threads
started.value:  105     threads
.
.
.
```

7.1.3 dumpMetrics

Use with WLST: Online

7.1.3.1 Description

Displays available metrics in the internal format or in XML. The returned value is a text document.

7.1.3.2 Syntax

```
dumpMetrics([servers,] [format])
```

Argument	Definition

servers	Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic server names and OPMN-managed component names.
	To specify one server, use the following syntax:
	`servers='servername'`
	To specify multiple servers, use one of the following syntax options:
	`servers=['servername1', 'servername2', ...]`
	`servers=('servername1', 'servername2', ...)`
	If this argument is not specified, the command returns the list of metric tables from all WebLogic servers and OPMN-managed components.

format	Optional. Specifies the command output format. Valid values are 'raw' (the default) and 'xml'. For example:
	`format='raw'`
	`format='xml'`
	DMS raw format is a simple metric display format; it displays one metric per line.

7.1.3.3 Example

The following example outputs all available metrics, including native WebLogic Server metrics and internal DMS metrics, in the XML format:

```
dumpMetrics(format='xml')
<table name='weblogic_j2eeserver:jvm' keys='ServerName serverName'
 componentId='bam_server1' cacheable='false'>
<row cacheable='false'>
<column name='serverName'><![CDATA[bam_server1]]></column>
<column name='nurserySize.value' type='DOUBLE'>0.0</column>
<column name='jdkVersion.value'><![CDATA[1.6.0_05]]></column>
<column name='jdkVendor.value'><![CDATA[BEA Systems, Inc.]]></column>
<column name='daemonThreads.active' type='LONG'>68</column>
```

```
<column name='cpuUsage.percentage' type='DOUBLE'>100.0</column>
<column name='threads.active' type='LONG'>71</column>
<column name='ServerName'><![CDATA[bam_server1]]></column>
<column name='heapUsed.value' type='DOUBLE'>0.0</column>
</row>
```

The following example outputs metrics from Server-0 in the default raw format:

```
dumpMetrics(servers='Server-0')
.
.
.
    /JVM/MxBeans/threads/Thread-44 [type=JVM_Thread]
     ECID.value:        null
     RID.value: null
     blocked.value:     0       msec
     blockedCount.value:        1       times
     cpu.value: 40      msecs
     lockName.value:    null
     lockOwnerID.value: null
     lockOwnerName.value:       null
     name.value:        LDAPConnThread-0 ldap://10.229.149.27:7001
     state.value:       RUNNABLE
     waited.value:      0       msec
     waitedCount.value: 0       times
    /JVM/MxBeans/threads/Thread-45 [type=JVM_Thread]
     ECID.value:        null
     RID.value: null
     blocked.value:     0       msec
.
.
.
```

The following example outputs metrics from Server-0 and Server-1 in XML format:

```
dumpMetrics(servers=['soa_server1', 'bam_server1'], format='xml')
<table name='oracle_soainfra:high_latency_sync_composites' keys='ServerName
 soainfra_composite soainfra_composite_revision soainfra_domain'
 componentId='bam_server1' cacheable='false'>
</table>
<table name='weblogic_j2eeserver:ejb_transaction' keys='ServerName appName
 ejbModuleName name serverName' componentId='bam_server1' cacheable='false'>
<row cacheable='false'>
<column name='serverName'><![CDATA[bam_server1]]></column>
<column name='name'><![CDATA[MessagingClientParlayX]]></column>
<column name='ejbTransactionCommit.percentage' type='DOUBLE'>0.0</column>
<column name='ejbTransactionRollback.completed' type='LONG'>0</column>
<column name='ejbTransactionTimeout.throughput' type='DOUBLE'>0.0</column>
<column name='ejbTransactionCommit.completed' type='LONG'>0</column>
<column name='ejbTransactionTimeout.completed' type='LONG'>0</column>
<column name='appName'><![CDATA[usermessagingserver]]></column>
<column name='ejbTransactionRollback.throughput' type='DOUBLE'>0.0</column>
<column name='ServerName'><![CDATA[bam_server1]]></column>
<column name='ejbTransactionCommit.throughput' type='DOUBLE'>0.0</column>
<column
name='ejbModuleName'><![CDATA[sdpmessagingclient-ejb-parlayx.jar]]></column>
</row>
.
.
.
```

7.1.4 reloadMetricRules

Use with WLST: Online

7.1.4.1 Description

Reloads the metric rules. You must run this command after you deploy OPMN-managed components or after you modify metric rules. Generally, Oracle does not recommend that you modify metric rules.

7.1.4.2 Syntax

```
reloadMetricRules([servers])
```

Argument	Definition

servers	Optional. Specifies the servers that should reload their metric rules. Valid values are a list of WebLogic server names and OPMN-managed component names.
	To specify one server, use the following syntax:
	`servers='servername'`
	To specify multiple servers, use the following syntax:
	`servers=['servername1', 'servername2', ...]`
	`servers=('servername1', 'servername2', ...)`
	If this argument is not specified, the command reloads the metric rules for all WebLogic servers and OPMN-managed components.

7.1.4.3 Example

The following example reloads metric rules at the specified Managed Server:

```
reloadMetricRules(servers='soa_server1')
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean
 as the root.
For more help, use help(domainRuntime)

loaded 'server-oracle_eps_server-11.0.xml'
loaded 'server-weblogic_j2eeserver-11.0.xml'
loaded 'server-oracle_bamweb-11.0.xml'
loaded 'server-oracle_federation-11.0.xml'
loaded 'server-portal-11.0.xml'
loaded 'server-weblogic_j2ee_application_webcenter-11.0.xml
.
.
.
```

8

Logging Custom WLST Commands

Use the logging commands to configure settings for log files and to view and search log files. Table 8–1 describes the different categories of logging commands.

For additional details about configuring and searching log files, see the chapter "Managing Log Files and Diagnostic Data" in the *Oracle Fusion Middleware Administrator's Guide*.

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Table 8–1 Logging Command Categories

Command category	Description
Log Configuration Commands	Configure settings for log files, such as the level of information written to the file or the maximum file size.
Search and Display Commands	View Oracle Fusion Middleware log files and search log files for particular messages.

8.1 Log Configuration Commands

Use the commands in Table 8–2 to configure settings for log files, such as the level of information written to the file or the maximum file size. In the Use with WLST column, online means the command can only be used when connected to a running server. Offline means the command can only be used when not connected to a running server. Online or offline means the command can be used in both situations.

Table 8–2 Logging Configuration Commands

Use this command...	To...	Use with WLST...
configureLogHandler	Configure an existing log handler, add a new handler, or remove existing handlers.	Online
getLogLevel	Get the level for a given logger.	Online
listLoggers	Get the list of loggers and the level of each logger.	Online
listLogHandlers	List the configuration of one of more log handlers.	Online
setLogLevel	Set the level for a given logger.	Online

8.1.1 configureLogHandler

Command Category: Log Configuration

Use with WLST: Online

8.1.1.1 Description

Configures an existing Java logging handler, adds a new handler, or removes an existing handler. It returns a java.util.List with one entry for each handler. Each entry is a javax.management.openmbean.CompositeData object describing the handler.

With this command, you can change the location of the log files, the frequency of the rotation of log files, and other log file properties.

8.1.1.2 Syntax

```
configureLogHandler(options)
```

Argument	Definition
options	Comma-separated list of options, specified as name-value pairs. Valid options include: ■ **target**—The name of a WebLogic server, or a string describing an OPMN-managed component. For OPMN-managed components, refer to the component's documentation for details. The default value is the server to which WLST is connected. ■ **name**—The name of a log handler. This option is required. ■ **maxFileSize**—The value of the maxFileSize attribute for an ODL handler. The value is a string representing a numeric value, optionally followed by a suffix indicating a size unit (k for kilobytes, m for megabytes, g for gigabytes). If you do not specify a suffix, the value is returned in bytes. ■ **maxLogSize**—The value of the maxLogSize attribute for an ODL handler. The value is a string representing a numeric value, optionally followed by a suffix indicating a size unit (k for kilobytes, m for megabytes, g for gigabytes). ■ **rotationFrequency**—The value of the rotation frequency for an ODL handler. The value is a string representing a numeric value, optionally followed by a suffix indicating a time unit (m for minutes, h for hours, d for days). The default unit is minutes. The following special values are also accepted and are converted to a numeric value in minutes: HOUR, HOURLY, DAY, DAILY, WEEK, WEEKLY, MONTH, MONTHLY. ■ **baseRotationTime**—The base rotation time, to be used with the rotationFrequency option. The value must be a string representing a date/time value. It can be a full date/time in ISO 8601 date/time format, or a short form including only hours and minutes. The default baseRotationTime is 00:00. ■ **retentionPeriod**—The amount of time, in minutes, that the log file is retained. The value must be a string representing a numeric value, optionally followed by a suffix indicating a time unit (m for minutes, h for hours, d for days). The default unit is minutes. The following special values are also accepted and are converted to a numeric value in minutes: HOUR, HOURLY, DAY, DAILY, WEEK, WEEKLY, MONTH, MONTHLY. ■ **format**—The format for the ODL handler. Valid values are one of the following strings: "ODL-Text" or "ODL-XML". The default format is ODL-Text. ■ **encoding**—The character encoding for the log file. ■ **path**—The log file path. ■ **handlerType**—The name of the Java class that provides the handler implementation. It must be an instance of java.util.logging.Handler or oracle.core.ojdl.logging.HandlerFactory.

Argument	Definition
options (continued)	■ **propertyName**—The name of an advanced handler property to be added or updated. The property value is specified with the propertyValue option. See the documentation for the handler for valid properties. ■ **propertyValue**—The new value for the handler property defined by the propertyName option. ■ **addProperty**—A Jython boolean value. Used in conjunction with the propertyName and propertyValue options to define that a new property is to be added to the handler. ■ **removeProperty**—A list of one or more handler properties to be removed. ■ **addHandler**—A boolean value. If the value is true, then the named handler will be added. ■ **removeHandler**—A boolean value. If the value is true, then the named handler is removed. ■ **addToLogger**—A list of logger names. The handler is added to the given logger names. ■ **removeFromLogger**—A list of logger names. The handler is removed from the given loggers.

8.1.1.3 Example

The following example specifies the maximum file size for the odl-handler:

```
configureLogHandler(name="odl-handler", maxFileSize="5M")
```

The following example specifies the rotation frequency for the odl-handler:

```
configureLogHandler(name="odl-handler", rotationFrequency="daily")
```

The following example specifies the rotation frequency and the retention period for the odl-handler. It also removes the properties maxFileSize and maxLogSize:

```
configureLogHandler(name="odl-handler", rotationFrequency="daily",
         retentionPeriod="week", removeProperty=['maxFileSize','maxLogSize'])
```

8.1.2 getLogLevel

Command Category: Log Configuration

Use with WLST: Online

8.1.2.1 Description

Returns the level of a given Java logger.

The returned value is a string with the logger's level, or None if the logger does not exist. An empty string indicates that the logger level is null.

8.1.2.2 Syntax

```
getLogLevel(options)
```

Argument	Definition
options	Comma-separated list of options, specified as name-value pairs. Valid options include: ■ **target**—The name of a WebLogic server, or a string describing an OPMN-managed component. For OPMN-managed components, refer to the component's documentation for details. The default value is the server to which WLST is connected. ■ **logger**—A logger name. An empty string denotes the root logger. This option is required and has no default. ■ **runtime**—A Jython boolean value (0 or 1) that determines if the operation is to list runtime loggers or config loggers. The default value is 1 (runtime).

8.1.2.3 Example

The following example returns the level for the logger oracle:

```
getLogLevel(logger='oracle')
```

The following example returns the level for the logger oracle, specifying only config loggers:

```
getLogLevel(logger='oracle', runtime=0)
```

The following example returns the level for the logger oracle on the Oracle WebLogic Server server2:

```
getLogLevel(logger='oracle', target='server2')
```

8.1.3 listLoggers

Command Category: Log Configuration

Use with WLST: Online

8.1.3.1 Description

Lists Java loggers and their levels. The command returns a PyDictionary object where the keys are logger names and the associated values are the logger levels. An empty level is used to indicate that the logger does not have the level set.

8.1.3.2 Syntax

```
listLoggers([options])
```

Argument	Definition
options	An optional comma-separated list of options, specified as name-value pairs. Valid options include: ■ **target**—The name of a WebLogic server, or a string describing an OPMN-managed component. For OPMN-managed components, refer to the component's documentation for details. The default value is the server to which WLST is connected. ■ **pattern**—A regular expression pattern that is used to filter logger names. The default value returns all logger names. ■ **runtime**—A Jython boolean value (0 or 1) that determines if the operation is to list runtime loggers or config loggers. The default value is 1.

8.1.3.3 Example

The following example lists all of the loggers:

```
listLoggers()
```

The following example lists all of the loggers that start with the name oracle.*.

```
listLoggers(pattern="oracle.*")
```

The following example list all config loggers:

```
listLoggers(runtime=0)
```

The following example list all loggers for the WebLogic server server1:

```
listLoggers(target="server1")
```

8.1.4 listLogHandlers

Command Category: Log Configuration

Use with WLST: Online

8.1.4.1 Description

Lists Java log handlers configuration. This command returns a java.util.List with one entry for each handler. Each entry is a javax.management.openmbean.CompositeData object describing the handler.

8.1.4.2 Syntax

```
listLogHandlers([options])
```

Argument	Definition
`options`	An optional comma-separated list of options, specified as name-value pairs. Valid options include: ■ **target**—The name of a WebLogic server, or a string describing an OPMN-managed component. For OPMN-managed components, refer to the component's documentation for details. The default value is the server to which WLST is connected. ■ **name**—The name of a log handler. If the name is not provided, then all handlers are listed.

8.1.4.3 Example

The following example lists all log handlers:

```
listLogHandlers()
```

The following example lists all log handlers named odl-handler:

```
listLogHandlers(name="odl-handler")
```

The following example lists all log handlers for the WebLogic server server1:

```
listLogHandlers(target="server1")
```

8.1.5 setLogLevel

Command Category: Log Configuration

Use with WLST: Online

8.1.5.1 Description

Sets the level of information written by a given Java logger to a log file.

8.1.5.2 Syntax

```
setLogLevel(options)
```

Argument	Definition
`options`	Comma-separated list of options, specified as name-value pairs. Valid options include: ■ **target**—The name of a WebLogic server, or a string describing an OPMN-managed component. For OPMN-managed components, refer to the component's documentation for details. The default value is the server to which WLST is connected. ■ **logger**—A logger name. An empty string denotes the root logger. This option is required and has no default. The command throws an exception if the logger does not exist, unless the addLogger option is also used. ■ **addLogger**—A Jython boolean value (0 or 1) that determines if the logger should be created if it does not exist. ■ **level**—The level name. It can be either a Java level or an ODL level. Some valid Java levels are: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, OR FINEST. Valid ODL levels include a message type followed by a colon and a message level. The valid ODL message types are: INCIDENT_ERROR, ERROR, WARNING, NOTIFICATION, TRACE, and UNKNOWN. The message level is represented by an integer value that qualifies the message type. Possible values are from 1 (highest severity) through 32 (lowest severity). This option is required; there is no default value. ■ **runtime**—A Jython boolean value (0 or 1) that determines if the operation is to list runtime loggers or config loggers. The default value is 1 (runtime). If the target is an OPMN-managed component that does not support changing runtime loggers, this option is ignored. ■ **persist**—A Jython boolean value (0 or 1) that determines if the level should be saved to the configuration file. The default value is 1.

8.1.5.3 Example

The following example sets the log level to NOTIFICATION:1 for the logger oracle.my.logger:

```
setLogLevel(logger="oracle.my.logger", level="NOTICATION:1")
```

The following example sets the log level to TRACE:1 for the logger oracle.my.logger and specifies that the level should be saved to the configuration file:

```
setLogLevel(logger="oracle.my.logger", level="TRACE:1", persist=0)
```

The following example sets the log level to WARNING for the config logger oracle.my.logger on the WebLogic server server1:

```
setLogLevel(target="server1", logger="oracle.my.logger", level="WARNING",
runtime=0)
```

8.2 Search and Display Commands

Use the commands in Table 8–3 to view Oracle Fusion Middleware log files and to search log files for particular messages.

Table 8–3 Search and Display Commands

Use this command...	To...	Use with WLST...
displayLogs	List the logs for one or more components.	Online or Offline
listLogs	Search and display the contents of log files.	Online or Offline

8.2.1 displayLogs

Command Category: Search and Display

Use with WLST: Online or Offline

8.2.1.1 Description

Search and display the contents of diagnostic log files. The command returns a value only when the returnData option is set to true. By default it will not return any data. The return value depends on the option used.

8.2.1.2 Syntax

```
displayLogs([searchString,][options])
```

Argument	Definition
`searchString`	An optional search string. Only messages that contain the given string (case-insensitive) will be returned. Note that the displayLogs command can read logs in multiple formats and it converts the messages to ODL format. The search will be performed in the native format, if possible. Otherwise, it may be performed in the message contents, and it may exclude mark-up. Therefore you should avoid using mark-up characters in the search string.
`options`	An optional comma-separated list of options, specified as name-value pairs. Valid options include: ■ **target**—The name of a WebLogic server, or an OPMN-managed component. For an OPMN managed component, the syntax for the target is "opmn:*instance-name*/*component-name*" In connected mode, the default target is the WebLogic domain. In disconnected mode, there is no default; the target option is required. ■ **oracleInstance**—In disconnected mode, defines the path to the, ORACLE_INSTANCE (or WebLogic domain home). In connected mode, this option is ignored. ■ **log**—A log file path. The command will read messages from the given log file. If the log file path is not given, the command will read all logs associated with the given target.

Argument	Definition
`options, continued`	■ **last**—An integer value. Restricts the search to messages logged within the last minutes. The value can have a suffix 's' (second), 'm' (minute), 'h' (hour), or 'd' (day) to specify a different time unit. (For example, last='2h' will be interpreted as the last 2 hours). ■ **tail**—An integer value. Restrict the search to the last *n* messages from each log file and limits the number of messages displayed to *n*. ■ **pattern**—A regular expression pattern. Only messages that contain the given pattern are returned. Using the pattern option is similar to using the searchString argument, except that you can use a regular expression. The regular expression pattern search is case sensitive (unless you explicitly turn on case-insensitive flags in the pattern). The pattern must follow java.util.regex syntax. ■ **ecid**—A string or string sequence containing one or more Execution Context ID (ECID) values to be used as a filter for log messages. ■ **component**—A string or string sequence containing one or more component ID values to be used as a filter for log messages. ■ **module**—A string or string sequence containing one or more module ID values to be used as a filter for log messages. ■ **type**—A string or string sequence containing one or more message type values to be used as a filter for log messages. ■ **app**—A string or string sequence containing one or more application values to be used as a filter for log messages. ■ **query**—A string that specifies an expression used to filter the contents of log messages. A simple expression has the form: `field-name operator value` where *field-name* is a log record field name and *operator* is an appropriate operator for the field type (for example, you can specify equals, startsWith, contains or matches for string fields). A field name is either one of the standard ODL attribute names (such as COMPONENT_ID, MSG_TYPE, MSG_TEXT, and SUPPL_DETAIL), or the name of a supplemental attribute (application specific), prefixed by SUPPL_ATTR. (for example, SUPPL_ATTR.myAttribute). A few common supplemental attributes can be used without the prefix, for example, you can use "APP" to filter by application name. You can combine multiple simple expressions using the boolean operators "and", "or" and "not" to create complex expressions, and you can use parenthesis for grouping expressions. See the *Oracle Fusion Middleware Administrator's Guide* for a detailed description of the query syntax. ■ **groupBy**—A string list. When the groupBy option is used, the output is a count of log messages, grouped by the attributes defined in the string list. ■ **orderBy**—A string list that defines the sort order for the result. The values are log message attribute names. The name may be extended with an optional suffix ":asc" or ":desc" to specify ascending or descending sorting. The default sort order is ascending. By default, the result is sorted by time.

Argument	Definition
options (continued)	■ **format**—A string defined the output format. Valid values are ODL-Text, ODL-XML, ODL-complete and simple. The default format is ODL-Text. ■ **exportFile**—The name of a file to where the command output is written. By default, the output is written to standard output. ■ **follow (f)**—Puts the command in "follow" mode so that it continues to read the logs and display messages as new messages are added to the logs (similar to the UNIX "tail -f" command). The command will not return when the f option is used. This option is currently not supported with OPMN-managed components. ■ **returnData**—A Jython boolean value (0 or 1). If the value is true the command will return data (for example, to be used in a script). The default value is false, which means that the command only displays the data but does not return any data.

8.2.1.3 Example

The following example displays the last 100 messages from all log files in the domain:.

```
displayLogs(tail=100)
```

The following example displays all messages logged in the last 15 minutes:

```
displayLogs(last='15m')
```

The following example displays log messages that contain a given string:

```
displayLogs('Exception')
```

The following example displays log messages that contain a given ECID:

```
displayLogs(ecid='0000Hl9TwKUCslT6uBi8UH18lkWX000002')
```

The following example displays log messages of type ERROR or INCIDENT_ERROR:

```
displayLogs(type=['ERROR','INCIDENT_ERROR'])
```

The following example displays log messages for a given Java EE application:

```
displayLogs(app="myApplication")
```

The following example displays messages for an OPMN-managed component:

```
displayLogs(target="opmn:instance1/ohs1")
```

The following example displays a message summary by component and type:

```
displayLogs(groupBy=['COMPONENT_ID', 'MSG_TYPE'])
```

The following example displays messages for a particular time interval:

```
displayLogs(query="TIME from 11:15 and TIME to 11:20")
```

The following example shows an advanced query:

```
displayLogs(query="TIME from 11:15 and TIME to 11:20 and ( MSG_TEXT contains
exception or SUPPL_DETAIL contains exception )")
```

A similar query could be written as:

```
displayLogs("exception", query="TIME from 11:15 and TIME to 11:20")
```

8.2.2 listLogs

Command Category: Search and Display

Use with WLST: Online or Offline

8.2.2.1 Description

Lists log files for Oracle Fusion Middleware components. This command returns a PyArray with one element for each log. The elements of the array are javax.management.openmbean.CompositeData objects describing each log.

8.2.2.2 Syntax

```
listLogs([options]
```

Argument	Definition
`options`	An optional comma-separated list of options, specified as name-value pairs. Valid options include: ■ **target**—The name of a WebLogic server, or an OPMN-managed Oracle Fusion Middleware component. For an OPMN-managed component, the syntax for the target is "opmn:*instance-name*/*component-name*" In connected mode, the default target is the WebLogic domain. In disconnected mode, there is no default; the target option is required. ■ **oracleInstance**—In disconnected mode, defines the path to the ORACLE_INSTANCE (or WebLogic domain home). In connected mode, this option is ignored. ■ **unit**—defines the unit to use for reporting file size. Valid values are B (bytes), K (kilobytes), M (megabytes), G (gigabytes), or H (display size in a human-readable form, similar to the UNIX "ls -h" option). The default value is H. ■ **fullTime**—A Jython Boolean value. If true, reports the full time for the log file last modified time. Otherwise, it displays a short version of the time. The default value is false.

8.2.2.3 Example

The following example lists all of the log files for the WebLogic domain:

```
listLogs()
```

The following example lists the log files for the WebLogic server server1:

```
listLogs(target="server1")
```

The following example lists the log files for the Oracle HTTP Server ohs1:

```
listLogs(target="opmn:instance1/ohs1")
```

The following example, used in disconnected mode, lists the log files for the WebLogic server server1:

```
listLogs(oracleInstance="/middleware/user_projects/domains/base_domain",
           target="server1")
```

9

Metadata Services (MDS) Custom WLST Commands

Use the Oracle Metadata Services (MDS) commands in the categories listed in Table 9–1 to manage Oracle Metadata Services (MDS).

For additional details about creating and managing an MDS repository, see the chapter "Managing the Oracle Metadata Repository" in the *Oracle Fusion Middleware Administrator's Guide*. For information about the roles needed to perform each operation, see "Understanding MDS Operations" in the *Oracle Fusion Middleware Administrator's Guide*.

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Table 9–1 MDS Command Categories

Command category	Description
Repository Management Commands	Manage the MDS repository.
Application Metadata Management Commands	Manage the application metadata in the MDS repository.
Application Label Management Commands	Manage the labels for the application.
Application Management Deployment Commands	Manage the application deployment.

9.1 Repository Management Commands

Use the MDS commands listed in Table 9–2 to manage the MDS repository. In the Use with WLST column, online means the command can only be used when connected to a running server. Offline means the command can only be used when not connected to a running server. Online or offline means the command can be used in both situations.

Table 9–2 Repository Management Commands

Use this command...	To...	Use with WLST...
createMetadataPartition	Create a metadata repository partition.	Online

Table 9–2 (Cont.) Repository Management Commands

Use this command...	To...	Use with WLST...
deleteMetadataPartition	Delete a metadata repository partition.	Online
deregisterMetadataDBRepository	Deregister a database-based MDS repository.	Online
registerMetadataDBRepository	Register a database-based MDS repository.	Online

9.1.1 createMetadataPartition

Command Category: Repository Management

Use with WLST: Online

9.1.1.1 Description

An application needs a metadata partition in the repository to manage its metadata. This command creates a partition with the given name in the specified repository. Each deployed application uses a logical partition in metadata repository. A metadata repository is used as a common repository for managing metadata of different applications. This logical partition also helps in maintaining the metadata lifecycle. Before deploying a application, you create a partition for it in MDS repository.

9.1.1.2 Syntax

Argument	Definition
repository	The name of the repository where the partition will be created.
partition	The name of the partition to create in the repository.

9.1.1.3 Example

The following example creates the metadata partition "part1" in repository "mds-myrepos":

```
wls:/weblogic/serverConfig> createMetadataPartition(repository='mds-myrepos',
                                partition='part1')
Executing operation: createMetadataPartition
Metadata partition created: part1
"part1"
wls:/weblogic/serverConfig>
```

9.1.2 deleteMetadataPartition

Command Category: Repository Management

Use with WLST: Online

9.1.2.1 Description

Deletes a metadata partition in the specified repository. When you delete a repository partition, all of the metadata in that partition is lost.

9.1.2.2 Syntax

```
deleteMetadataPartition(repository, partition)
```

Argument	Definition
repository	The name of the repository that contains the partition.
partition	The name of the partition to delete in the repository.

9.1.2.3 Example

The following example deletes the metadata partition "part1" from the repository "mds-myrepos":

```
wls:/weblogic/serverConfig> deleteMetadataPartition(repository='mds-myrepos',
                                partition='part1')
Executing operation: deleteMetadataPartition
Metadata partition deleted: part1
wls:/weblogic/serverConfig>
```

9.1.3 deregisterMetadataDBRepository

Command Category: Repository Management

Use with WLST: Online

9.1.3.1 Description

Removes the database metadata repository registration as a System JDBC data source in the domain. After this command completes successfully, applications can no longer use this repository.

9.1.3.2 Syntax

```
deregisterMetadataDBRepository(name)
```

Argument	Definition
name	The name of the repository to deregister.

9.1.3.3 Example

The following example deregisters the metadata repository "mds-myrepos":

```
wls:/weblogic/serverConfig> deregisterMetadataDBRepository('mds-myrepos')
Executing operation: deregisterMetadataDBRepository.
Metadata DB repository "mds-myrepos" was deregistered successfully.
wls:/weblogic/serverConfig>
```

9.1.4 registerMetadataDBRepository

Command Category: Repository Management

Use with WLST: Online

9.1.4.1 Description

A database metadata repository should be registered with WebLogic servers before the application can use it. This command registers a System JDBC data source with the domain for use as database-based metadata repository.

9.1.4.2 Syntax

```
registerMetadataDBRepository(name, dbVendor, host, port, dbName, user, password,
[targetServers])
```

Argument	Definition
name	The name of the repository to register.
dbVendor	The database vendor. The acceptable values are ORACLE or MSSQL.
host	The host name or the IP address of the database.
port	The port number used by the database.
dbName	The service name of the database. For example, orcl.*hostname*.com
user	The database user name.
password	The password for the database user.
targetServers	Optional. The WebLogic servers to which this repository will be registered. If this argument is not specified, then the repository will be registered only to the Administration Server. To specify multiple servers, separate the names with a comma.

9.1.4.3 Example

The following example registers the metadata repository `myrepos`, to two servers, with the database parameters:

```
wls:/weblogic/serverConfig> registerMetadataDBRepository('myrepos','ORACLE',
              'test.oracle.com','1521','mds', 'user1','x','server1, server2')
Executing operation: registerMetadataDBRepository.
Metadata DB repository "mds-myrepos" was registered successfully.
'mds-myrepos'
wls:/weblogic/serverConfig>
```

9.2 Application Metadata Management Commands

Use the commands in Table 9–3 to manage application metadata.

Table 9–3 Application Metadata Commands

Use this command...	To...	Use with WLST...
deleteMetadata	Deletes the metadata in the application repository.	Online
exportMetadata	Exports metadata for an application.	Online
importMetadata	Imports metadata for an application.	Online
purgeMetadata	Purge metadata.	Online

9.2.1 deleteMetadata

Command Category: Application Metadata

Use with WLST: Online

9.2.1.1 Description

Deletes the selected documents from the application repository. When this command is run against repositories that support versioning, that is a database-based repository,

delete is logical and marks the tip version (the latest version) of the selected documents as "deleted" in the MDS repository partition.

You may want to delete metadata when the metadata is moved from one repository to another. In such a case, after you have exported the metadata, you can delete the metadata in the original repository.

9.2.1.2 Syntax

```
deleteMetadata(application, server, [docs], [restrictCustTo], [excludeAllCust],
 [excludeBaseDocs], [excludeExtendedMetadata], [cancelOnException],
[applicationVersion]
```

Argument	Definition
application	The name of the application for which the metadata is to be deleted.
server	The target server on which this application is deployed.
docs	Optional. A list of comma-separated, fully qualified document names or document name patterns, or both. The patterns can have the following wildcard characters: * and **. The "*" represents all documents under the current namespace. The"**" represents all documents under the current namespace and also recursively includes all documents in sub-namespaces. For example, "/oracle/*" will include all documents under "/oracle/" but not include documents under "/oracle/mds/". As another example, "/oracle/**" will include all documents under "/oracle/" and also under "/oracle/mds/" and any other documents further in the namespace chain.
restrictCustTo	Optional. A list of comma-separated customization layer names used to restrict the delete operation so that it deletes only customization documents that match the specified customization layers. This argument will be ignored if the excludeAllCust argument is also specified.
excludeAllCust	Optional. A Boolean value (true or false) that specifies whether or not to delete all customization documents. This argument defaults to false. It overrides the restrictCustTo option.
excludeBaseDocs	Optional. A Boolean value (true or false) that specifies whether or not to delete base documents. This argument defaults to false.
excludeExtendedMetadata	Optional. A Boolean value (true or false) that specifies whether or not to delete the Extended Metadata documents. This argument defaults to false.
cancelOnException	Optional. A Boolean value (true or false) that specifies whether or not to abort the delete operation when an exception is encountered. On abort, the delete is rolled back if that is supported by the target store. This argument defaults to true.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

9.2.1.3 Examples

The following example deletes metadata files under the package "mypackage" from sampleApp deployed in the server "srg":

```
wls:/weblogic/serverConfig> deleteMetadata(application='mdsapp',
          server='srg', docs='/mypackage/*')
Executing operation: deleteMetadata.
```

```
"deleteMetadata" operation completed. Summary of "deleteMetadata" operation is:
List of documents successfully deleted:
/mypackage/jobs.xml
/mypackage/mo.xml
2 documents successfully deleted.
wls:/weblogic/serverConfig>
```

The following example deletes metadata files under the package "mypackage" from sampleApp deployed in the server "srg" and specifies to exclude extended metadata and all customizations:

```
wls:/weblogic/serverConfig> deleteMetadata(application='mdsapp',
        server='srg', docs='/mypackage/*', cancelOnException='false',
        excludeExtendedMetadata='true',
        excludeAllCust='true')
Executing operation: deleteMetadata.
"deleteMetadata" operation completed. Summary of "deleteMetadata" operation is:
List of documents successfully deleted:
/mypackage/jobs.xml
/mypackage/mo.xml
2 documents successfully deleted.
wls:/weblogic/serverConfig>
```

9.2.2 exportMetadata

Command Category: Application Metadata

Use with WLST: Online

9.2.2.1 Description

The application metadata can be transferred from one server location (for example, testing) to another server location (for example, production) by exporting and importing the metadata.

Use this command to export metadata.

9.2.2.2 Syntax

```
exportMetadata(application, server, toLocation, [docs,] [restrictCustTo],
 [excludeAllCust], [excludeBaseDocs], [excludeExtendedMetadata], [fromLabel],
 [toLabel], [applicationVersion])
```

Argument	Definition
application	The name of the application for which the metadata is to be exported.
server	The target server on which this application is deployed.
toLocation	The target directory to which documents selected from the source partition will be transferred. The directory must be a local or network directory where the application is physically deployed. This argument can be used as temporary file system for transferring metadata from one server to another.

Argument	Definition
docs	Optional. A list of comma-separated, fully qualified document names or document name patterns, or both. The patterns can have the following wildcard characters: * and **. This argument defaults to "/**". The "*" represents all documents under the current namespace. The"**" represents all documents under the current namespace and also recursively includes all documents in sub-namespaces. For example, "/oracle/*" will include all documents under "/oracle/" but not include documents under "/oracle/mds/". "/oracle/**" will include all documents under "/oracle/" and also under "/oracle/mds/" and any other documents further in the namespace chain.
restrictCustTo	Optional. A list of comma-separated customization layer names used to restrict the export operation to export only customization documents that match the specified customization layers. This argument will be ignored if the excludeAllCust argument is also specified.
excludeAllCust	Optional. A Boolean value (true or false) that specifies whether or not to export all customization documents. This argument defaults to false. This argument overrides the restrictCustTo argument.
excludeBaseDocs	Optional. A Boolean value (true or false) that specifies whether or not to export base documents. This argument defaults to false.
excludeExtendedMetadata	Optional. A Boolean value (true or false) that specifies whether or not to export the Extended Metadata documents. This argument defaults to false.
fromLabel	Optional. Transfers the documents from the source partition that is associated with this label.
toLabel	Optional. Works with the fromLabel argument to transfer the delta between fromLabel to toLabel from the source partition.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

9.2.2.3 Examples

The following example exports all metadata files from the application "mdsapp" deployed in the server "srg".

```
wls:/weblogic/serverConfig> exportMetadata(application='mdsapp',
                            server='srg',toLocation='/tmp/myrepos',docs='/**')
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean
as the root.
For more help, use help(domainRuntime)
Executing operation: exportMetadata.
"exportMetadata" operation completed. Summary of "exportMetadata" operation is:
List of documents successfully transferred:
/mypackage/write.xml
/mypackage/write1.xml
/sample1.jspx
3 documents successfully transferred.
wls:/weblogic/serverConfig>
```

The following example exports only the customization documents under the layer `user` without any base documents from label `label1` to label `label2`:

```
wls:/weblogic/serverConfig> exportMetadata(application='mdsapp',
```

```
                        server='srg',toLocation='/tmp/myrepos',
                        restrictCustTo='user',
                        excludeBaseDocs='true',
                        fromLabel='label1',
                        toLabel='label2',
                        applicationVersion='11.1.1')
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean
as the root.
For more help, use help(domainRuntime)
Executing operation: exportMetadata.
"exportMetadata" operation completed. Summary of "exportMetadata" operation is:
List of documents successfully transferred:
/mypackage/mdssys/cust/user/user1/write1.xml.xml
/mypackage/mdssys/cust/user/user2/write2.xml.xml
/sample1.jspx
3 documents successfully transferred.
```

9.2.3 importMetadata

Command Category: Application Metadata

Use with WLST: Online

9.2.3.1 Description

The application metadata can be transferred from one server location (for example, testing) to another server location (for example, production) by exporting and importing the metadata.

Use this command to import metadata.

9.2.3.2 Syntax

```
importMetadata(application, server, fromLocation, [docs,] [restrictCustTo],
    [excludeAllCust], [excludeBaseDocs], [excludeExtendedMetadata],
    [cancelOnException], [applicationVersion])
```

Argument	Definition
application	The name of the application for which the metadata is to be imported.
server	The target server on which this application is deployed.
fromLocation	The source directory from which documents will be selected for transfer. The directory must be a local or network directory where the application is physically deployed. This argument can be used as a temporary file system location for transferring metadata from one server to another.
docs	Optional. A list of comma-separated, fully qualified document names or document name patterns, or both. The patterns can have the following wildcard characters: * and **. This argument defaults to "/**". The "*" represents all documents under the current namespace. The"**" represents all documents under the current namespace and also recursively includes all documents in sub-namespaces. For example, "/oracle/*" will include all documents under "/oracle/" but not include documents under "/oracle/mds/". "/oracle/**" will include all documents under "/oracle/" and also under "/oracle/mds/" and any other documents further in the namespace chain.

Argument	Definition
restrictCustTo	Optional. A list of comma-separated customization layer names used to restrict the import operation to import only customization documents that match the specified customization layers. This argument will be ignored if the excludeAllCust argument is also specified.
excludeAllCust	Optional. A Boolean value (true or false) that specifies whether or not to import all customization documents. This argument defaults to false. This argument overrides the restrictCustTo argument.
excludeBaseDocs	Optional. A Boolean value (true or false) that specifies whether or not to import base documents. This argument defaults to false.
excludeExtendedMetadata	Optional. A Boolean value (true or false) that specifies whether or not to import the Extended Metadata documents. This argument defaults to false.
cancelOnException	Optional. A Boolean value (true or false) that specifies whether or not to abort the import operation when an exception is encountered.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

9.2.3.3 Example

The following example imports all metadata available in '/tmp/myrepos' to the application "mdsapp" deployed in the server "srg":

```
wls:/weblogic/serverConfig> importMetadata(application='mdsapp', server='srg',
                            fromLocation='/tmp/myrepos',docs="/**")
Executing operation: importMetadata.
"importMetadata" operation completed. Summary of "importMetadata" operation is:
List of documents successfully transferred:
/app1/jobs.xml
/app1/mo.xml
2 documents successfully transferred.
wls:/weblogic/serverConfig>
```

9.2.4 purgeMetadata

Command Category: Application Metadata

Use with WLST: Online

9.2.4.1 Description

Purges the unlabeled document's version from the application's repository. All documents will be purged if they are expired, based on Time-To-Live (the olderThan argument). Document versions that are attached to a label and those which are tip (latest version) are not purged. This command is applicable only for repositories that support versioning, that is, a database-based repository.

9.2.4.2 Syntax

```
purgeMetadata(application, server, olderThan, [applicationVersion])
```

Argument	Definition
application	The name of the application, used to identify the partition in the repository on which the purge operation will be run.
server	The target server on which this application is deployed.

Argument	Definition
olderThan	Document versions that are older than this value will be purged.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

9.2.4.3 Example

The following example purges the document version history for the application "mdsapp" deployed in the server "srg," if the version is older than 10 seconds:

```
wls:/weblogic/serverConfig> purgeMetadata('mdsapp', 'srg', 10)
Executing operation: purgeMetadata.
Metadata purged:Total number of versions: 10.
Number of versions purged: 0.
wls:/weblogic/serverConfig>
```

9.3 Application Label Management Commands

Use the commands in Table 9–4 to manage labels for applications.

Table 9–4 Application Label Management Commands

Use this command...	To...	Use with WLST...
createMetadataLabel	Creates a metadata label.	Online
deleteMetadataLabel	Deletes a metadata label from the repository partition	Online
listMetadataLabels	Lists metadata labels in the repository partition.	Online
promoteMetadataLabel	Promotes the metadata associated with a label to tip.	Online

9.3.1 createMetadataLabel

Command Category: Application Label Management

Use with WLST: Online

9.3.1.1 Description

Creates a new label for the documents in the application's repository partition. This command is applicable only for repositories that support versioning.

9.3.1.2 Syntax

```
createMetadataLabel(application, server, name, [applicationVersion])
```

Argument	Definition
application	The name of the application for which a label will be created in the partition configured for this application.
server	The target server on which this application is deployed.
name	The name of the label to create in the repository partition.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

9.3.1.3 Example

The following example creates the label "labe11" for the application "mdsapp" deployed in the server "srg":

```
wls:/weblogic/serverConfig> createMetadataLabel('mdsapp','srg','label1')
Executing operation: createMetadataLabel.
Created metadata label "label1".
wls:/weblogic/serverConfig>
```

9.3.2 deleteMetadataLabel

Command Category: Application Label Management

Use with WLST: Online

9.3.2.1 Description

Deletes a label for the documents in the application's repository partition. This command is applicable only for repositories that support versioning.

9.3.2.2 Syntax

```
deleteMetadataLabel(application, server, name, [applicationVersion])
```

Argument	Definition
application	The name of the application from whose associated partition the label is to be deleted.
server	The target server on which this application is deployed.
name	The name of the label to delete in the repository partition.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

9.3.2.3 Example

The following example deletes the metadata label "label1" from the application "mdsapp" deployed in the server "srg":

```
wls:/weblogic/serverConfig> deleteMetadataLabel('mdsapp','srg','label1')
Executing operation: deleteMetadataLabel.
Deleted metadata label "label1".
wls:/weblogic/serverConfig>
```

9.3.3 listMetadataLabels

Command Category:

Use with WLST: Online

9.3.3.1 Description

Lists all of the metadata labels in the application's repository partition. This command is applicable only for repositories that support versioning.

9.3.3.2 Syntax

```
listMetadataLabels(application, server, [applicationVersion])
```

Argument	Definition
application	The name of the application for which all of the labels in the repository partition associated with this application will be deleted.
server	The target server on which this application is deployed.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

9.3.3.3 Example

The following example lists the metadata labels available for the application "mdsapp" deployed in the server "srg":

```
wls:/weblogic/serverConfig> listMetadataLabels('mdsapp', 'srg')
Executing operation: listMetadataLabels.
Database Repository partition contains the following labels:
label2
label3
wls:/weblogic/serverConfig>
```

9.3.4 promoteMetadataLabel

Command Category: Application Label Management

Use with WLST: Online

9.3.4.1 Description

Promotes documents associated with a label to the tip version in the repository. This command is useful to achieve rollback capability. This command is applicable only for repositories that support versioning.

9.3.4.2 Syntax

```
promoteMetadataLabel(application, server, name, [applicationVersion])
```

Argument	Definition
application	The name of the application in whose associated repository the metadata is to be promoted to tip.
server	The target server on which this application is deployed.
name	The name of the label to promote in the repository partition.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

9.3.4.3 Example

The following example promotes the metadata label "label1" to tip in the application "mdsapp" deployed in the server "srg":

```
wls:/weblogic/serverConfig> promoteMetadataLabel('mdsapp', 'srg','label1')
Executing operation: promoteMetadataLabel.
Promoted metadata label "label1" to tip.
wls:/weblogic/serverConfig>
```

9.4 Application Management Deployment Commands

Use the commands in Table 9–5 to manage deployment.

Table 9–5 Application Management Deployment Commands

Use this command...	To...	Use with WLST...
getMDSArchiveConfig	Returns an MDSArchiveConfig object.	Offline
importMAR	Imports a MAR.	Online

9.4.1 getMDSArchiveConfig

Command Category: Application Management Deployment

Use with WLST: Offline

9.4.1.1 Description

Returns a handle to the MDSArchiveConfig object for the specified archive. The returned MDSArchiveConfig object's methods can be used to change application and shared repository configuration in an archive.

The MDSArchiveConfig object provides the following methods:

- **setAppMetadataRepository**—This method sets the connection details for the application metadata repository.

  If the archive's existing adf-config.xml file does not have any configuration for the application's metadata repository, then you must provide all necessary arguments to define the target repository. To define a database-based repository, provide the repository, partition, type, and jndi arguments. For a file-based repository, provide the path argument instead of jndi.

  If the adf-config.xml file already contains some configuration for the application's metadata repository, you can provide only a subset of arguments that you want to change. You do not need to provide all arguments in such a case. However, if the store type is changed, then the corresponding jndi or path argument is required.

- **setAppSharedMetadataRepository**—This method sets the connection details for the shared repository in the application archive that is mapped to specified namespace.

  If the archive's existing adf-config.xml file does not have any configuration for a shared metadata repository mapped to the specified namespace, you must provide all required arguments (in this case, repository, partition, type, and jndi or path). For a database-based repository, provide the jndi argument. For a file-based repository, path is a required argument.

  If the adf-config.xml file already contains some configuration for a shared metadata repository mapped to the specified namespace and you want to change some specific arguments, you can provide only a subset of those arguments; all others are not needed.

- **save**—If you specify the toLocation, then the changes will be stored in the target archive file and the original file will remain unchanged. Otherwise, the changes will be saved in the original file itself.

9.4.1.2 Syntax

```
archiveConfigObject = getMDSArchiveConfig(fromLocation)
```

Argument	Definition
fromLocation	The name of the ear file, including its complete path.

The syntax for setAppMetadataRepository is:

```
archiveConfigObject.setAppMetadataRepository([repository], [partition], [type],
[jndi], [path])
```

Argument	Definition
repository	Optional. The name of the application's repository.
partition	Optional. The name of the partition for the application's metadata.
type	Optional. The type of connection, file or database, to the repository. Valid values are 'File' or 'DB' (case insensitive).
jndi	Optional. The JNDI location for the database connection. This argument is required if the type is set to DB. This argument will not be considered if the type is set to File.
path	Optional. The location of the file metadata store. This argument is required if the type is set to File. This argument will not be considered if the type is set to DB.

The syntax for setAppSharedMetadataRepository is:

```
archiveConfigObject.setAppSharedMetadataRepository(namespace, [repository],
[partition], [type], [jndi], [path])
```

Argument	Definition
namespace	The namespace used for looking up the shared repository to set connection details.
repository	Optional. The name of the application's shared repository.
partition	Optional. The name of the partition for the application's shared metadata.
type	Optional. The type of connection, file or database, to the repository. Valid values are 'File' or 'DB' (case insensitive).
jndi	Optional. The JNDI location for the database connection. This argument is required if the type is set to DB. This argument will not be considered if the type is set to File.
path	Optional. The location of the file metadata store. This argument is required if the type is set to File. This argument will not be considered if the type is set to DB.

The syntax for save is:

```
archiveConfigObject.save([toLocation])
```

Argument	Definition
toLocation	Optional. The file name along with the absolute path to store the changes. If this option is not provided, the changes are written to the archive represented by this configuration object.

9.4.1.3 Examples

In the following example, if the adf-config.xml file in the archive does not have the application and shared metadata repositories defined, then you should provide the complete connection information.

```
wls:/offline> archive = getMDSArchiveConfig(fromLocation='/tmp/testArchive.ear')

wls:/offline> archive.setAppMetadataRepository(repository='AppRepos1',
        partition='partition1', type='DB', jndi='mds-jndi1')

wls:/offline> archive.setAppSharedMetadataRepository(namespace='/a',
        repository='SharedRepos1', partition='partition2', type='File',
        path='/temp/dir')
wls:/offline> archive.save()
```

In the following example, if the adf-config.xml file in the archive already has the application and shared metadata repositories defined, all arguments are optional. You can set only the arguments you want to change.

```
wls:/offline> archive = getMDSArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setAppMetadataRepository(partition='MDS-partition2')
wls:/offline> archive.setAppSharedMetadataRepository(namespace='/a',
         repository='SharedRepos2')
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

9.4.2 importMAR

Command Category: Application Management Deployment

Use with WLST: Online

9.4.2.1 Description

Imports the metadata from the MAR packaged along with the application's EAR file.

9.4.2.2 Syntax

```
importMAR(application, server, [applicationVersion])
```

Argument	Definition
application	The name of the application for which the metadata is to be imported.
server	The target server on which this application is deployed.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

9.4.2.3 Example

The following example imports metadata from the MAR to the application "mdsapp":

```
wls:/weblogic/serverConfig> importMAR('mdsapp','srg')
Executing operation: importMAR.
"importMAR" operation completed. Summary of "importMAR" operation is:
/app1/jobs.xml
/app1/mo.xml
2 documents successfully transferred.
wls:/weblogic/serverConfig>
```

10

Oracle SOA Suite Custom WLST Commands

This chapter describes WSLT commands for Oracle SOA Suite. These commands enable you to use WLST to configure SOA composite applications.

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide*.

This chapter includes the following sections:

For additional details about deployment, configuration plans, and test suites, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

10.1 Overview of WSLT Command Categories

WLST commands are divided into the categories shown in Table 10–1.

Table 10–1 Oracle SOA Suite Command Categories

Command category	Description
Section 10.2, "Deployment Commands"	Deploy and undeploy SOA composite applications.
Section 10.3, "SOA Composite Application Management Commands"	Start, stop, activate, retire, assign a default revision version, and list deployed SOA composite applications.

Table 10–1 (Cont.) Oracle SOA Suite Command Categories

Command category	Description
Section 10.4, "Configuration Plan Management Commands"	Attach, extract, generate, and validate configuration plans for SOA composite applications.
Section 10.5, "Task Validation Commands"	Validate human workflow tasks.
Section 10.6, "SOA Composite Application Compilation Commands"	Compile SOA composite applications.
Section 10.7, "SOA Composite Application Packaging Commands"	Package SOA composite applications into archive files to deploy.
Section 10.8, "SOA Composite Application Test Commands"	Test SOA composite applications prior to deployment in a production environment.

10.2 Deployment Commands

Use the deployment commands, listed in Table 10–2, to deploy and undeploy SOA composite applications.

Table 10–2 Deployment Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_deployComposite	Deploy a SOA composite application.	Offline
sca_undeployComposite	Undeploy a SOA composite application.	Offline

10.2.1 sca_deployComposite

Command Category: Deployment Commands

Use with WLST: Offline

10.2.1.1 Description

Deploys a SOA composite application to the Oracle WebLogic Server. This command does *not* package the artifact files of the application for deployment. See Section 10.7, "SOA Composite Application Packaging Commands" for instructions on packaging a SOA composite application.

10.2.1.2 Syntax

```
sca_deployComposite(serverURL, sarLocation, overwrite, user, password,
forceDefault, configplan)
```

Argument	Definition
serverURL	URL of the server that hosts the SOA Infrastructure application (for example, `http://myhost10:7001`).

Argument	Definition
sarLocation	Absolute path to one the following: ■ SOA archive (SAR) file. ■ ZIP file that includes multiple SARs, metadata archives (MARs), or both. ■ Enterprise archive (EAR) file that contains a SAR file.
overwrite	Optional. Indicates whether to overwrite an existing SOA composite application file. ■ `false` (default): Does not overwrite the file. ■ `true`: Overwrites the file.
user	Optional. User name to access the composite deployer servlet when basic authentication is configured.
password	Optional. Password to access the composite deployer servlet when basic authentication is configured.
forceDefault	Optional. Indicates whether to set the new composite as the default. ■ `true` (default): Makes it the default composite. ■ `false`: Does not make it the default composite.
configplan	Absolute path of a configuration plan to be applied to a specified SAR file or to all SAR files included in the ZIP file.

10.2.1.3 Examples

The following example deploys the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost10:7001",
"/tmp/sca_HelloWorld_rev1.0.jar")
```

The following example deploys the `HelloWorld` application as the default version.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost10:7001",
"/tmp/sca_HelloWorld_rev1.0.jar", true)
```

The following example deploys the `HelloWorld` application with a required user name when basic authentication is configured. You are then prompted to provide the password for this user name.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost10:7001",
"/tmp/sca_HelloWorld_rev1.0.jar", user="weblogic")
Password:
```

The following example deploys the `HelloWorld` application and applies the configuration plan named `deployplan.xml`.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost10:7001",
"/tmp/sca_HelloWorld_rev1.0.jar", forceDefault=false,
configplan="/tmp/deployplan.xml")
```

The following example deploys the `HelloWorld` ZIP file, which can include multiple SARs, MARs, or both.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://myhost:7001",
"/tmp/HelloWorld.zip")
```

10.2.2 sca_undeployComposite

Command Category: Deployment Commands

Use with WLST: Offline

10.2.2.1 Description

Undeploys a currently deployed SOA composite application.

10.2.2.2 Syntax

```
sca_undeployComposite(serverURL, compositeName, revision, user, password)
```

Argument	Definition
serverURL	URL of the server that hosts the SOA Infrastructure application (for example, `http://myhost10:7001`).
compositeName	Name of the SOA composite application.
revision	Revision ID of the SOA composite application.
user	Optional. User name to access the composite deployer servlet when basic authentication is configured.
password	Optional. Password to access the composite deployer servlet when basic authentication is configured.

10.2.2.3 Examples

The following example undeploys the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_undeployComposite("http://myhost10:7001",
"HelloWorld", "1.0")
```

The following example undeploys the `HelloWorld` application with a required user name when basic authentication is configured. You are then prompted to provide the password for this user name.

```
wls:/mydomain/ServerConfig> sca_undeployComposite("http://myhost10:7001",
"HelloWorld", "1.0", user="weblogic")
Password:
```

10.3 SOA Composite Application Management Commands

Use the management commands, listed in Table 10–3, to start, stop, activate, retire, assign a default revision version, and list deployed SOA composite applications.

Table 10–3 SOA Composite Application Management Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_startComposite	Start a previously stopped SOA composite application.	Offline
sca_stopComposite	Stop a SOA composite application.	Offline
sca_activateComposite	Activate a previously retired SOA composite application.	Offline
sca_retireComposite	Retire a SOA composite application.	Offline

Table 10–3 (Cont.) SOA Composite Application Management Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_assignDefaultComposite	Assign the default revision version to a SOA composite application.	Offline
sca_listDeployedComposites	List the deployed SOA composite applications.	Offline

10.3.1 sca_startComposite

Command Category: Application Management Commands

Use with WLST: Offline

10.3.1.1 Description

Starts a previously stopped SOA composite application.

10.3.1.2 Syntax

```
sca_startComposite(host, port, user, password, compositeName, revision, label)
```

Argument	Definition
host	Hostname of the Oracle WebLogic Server (for example, `myhost`).
port	Port of the Oracle WebLogic Server (for example, `7001`).
user	User name for connecting to the running server to get mBean information (for example, `weblogic`).
password	Password for the user name.
compositeName	Name of the SOA composite application.
revision	Revision of the SOA composite application.
label	Optional. Label of the SOA composite application. The label identifies the metadata service (MDS) artifacts associated with the application. If the label is not specified, the system finds the latest one.

10.3.1.3 Example

The following example starts revision `1.0` of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_startComposite("myhost", "7001", "weblogic",
"weblogic", "HelloWorld", "1.0")
```

10.3.2 sca_stopComposite

Command Category: Application Management Commands

Use with WLST: Offline

10.3.2.1 Description

Stops a currently running SOA composite application.

10.3.2.2 Syntax

```
sca_stopComposite(host, port, user, password, compositeName, revision, label)
```

Argument	Definition
host	Hostname of the Oracle WebLogic Server (for example, `myhost`).
port	Port of the Oracle WebLogic Server (for example, `7001`).
user	User name for connecting to the running server to get mBean information (for example, `weblogic`).
password	Password for the user name.
compositeName	Name of the SOA composite application.
revision	Revision of the SOA composite application.
label	Optional. Label of the SOA composite application. The label identifies the MDS artifacts associated with the application. If the label is not specified, the system finds the latest one.

10.3.2.3 Example

The following example stops revision `1.0` of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_stopComposite("myhost", "7001", "weblogic",
"weblogic", "HelloWorld", "1.0")
```

10.3.3 sca_activateComposite

Command Category: Application Management Commands

Use with WLST: Offline

10.3.3.1 Description

Activates a retired SOA composite application and its instances. You can then create new instances.

10.3.3.2 Syntax

```
sca_activateComposite(host, port, user, password, compositeName, revision, label)
```

Argument	Definition
host	Hostname of the Oracle WebLogic Server (for example, `myhost`).
port	Port of the Oracle WebLogic Server (for example, `7001`).
user	User name for connecting to the running server to get mBean information (for example, `weblogic`).
password	Password for the user name.
compositeName	Name of the SOA composite application.
revision	Revision of the SOA composite application.
label	Optional. Label of the SOA composite application. The label identifies the MDS artifacts associated with the application. If the label is not specified, the system finds the latest one.

10.3.3.3 Example

The following example activates revision `1.0` of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_activateComposite("myhost", "7001", "weblogic",
"weblogic", "HelloWorld", "1.0")
```

10.3.4 sca_retireComposite

Command Category: Application Management Commands

Use with WLST: Offline

10.3.4.1 Description

Stops and retires a SOA composite application and all its running instances. If the process life cycle is retired, you cannot create a new instance. Existing instances are allowed to complete normally.

10.3.4.2 Syntax

```
sca_retireComposite(host, port, user, password, compositeName, revision, label)
```

Argument	Definition
host	Hostname of the Oracle WebLogic Server (for example, `myhost`).
port	Port of the Oracle WebLogic Server (for example, `7001`).
user	User name for connecting to the running server to get mBean information (for example, `weblogic`).
password	Password for the user name.
compositeName	Name of the SOA composite application.
revision	Revision of the SOA composite application.
label	Optional. Label of the SOA composite application. The label identifies the MDS artifacts associated with the application. If the label is not specified, the system finds the latest one.

10.3.4.3 Example

The following example retires revision `1.0` of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_retireComposite("myhost", "7001", "weblogic",
"weblogic", "HelloWorld", "1.0")
```

10.3.5 sca_assignDefaultComposite

Command Category: Application Management Commands

Use with WLST: Offline

10.3.5.1 Description

Sets a SOA composite application revision as the default version. This revision is instantiated when a new request comes in.

10.3.5.2 Syntax

```
sca_assignDefaultComposite(host, port, user, password, compositeName, revision)
```

Argument	Definition
host	Hostname of the Oracle WebLogic Server (for example, `myhost`).
port	Port of the Oracle WebLogic Server (for example, `7001`).
user	User name for connecting to the running server to get mBean information (for example, `weblogic`).

Argument	Definition
password	Password for the user name.
compositeName	Name of the SOA composite application.
revision	Revision of the SOA composite application.

10.3.5.3 Example

The following example sets revision `1.0` of the `HelloWorld` application as the default version.

```
wls:/mydomain/ServerConfig> sca_assignDefaultComposite("myhost", "7001",
"weblogic", "weblogic", "HelloWorld", "1.0")
```

10.3.6 sca_listDeployedComposites

Command Category: Application Management Commands

Use with WLST: Offline

10.3.6.1 Description

Lists all SOA composite applications deployed to the SOA platform.

10.3.6.2 Syntax

```
sca_listDeployedComposites(host, port, user, password)
```

Argument	Definition
host	Hostname of the Oracle WebLogic Server (for example, `myhost`).
port	Port of the Oracle WebLogic Server (for example, `7001`).
user	User name for connecting to the running server to get mBean information (for example, `weblogic`).
password	Password for the user name.

10.3.6.3 Example

The following example lists all the deployed SOA composite applications on the server `myhost`.

```
wls:/mydomain/ServerConfig> sca_listDeployedComposites('myhost', '7001',
'weblogic', 'weblogic')
```

10.4 Configuration Plan Management Commands

Use the configuration plan management commands, listed in Table 10–4, to attach, extract, generate, and validate configuration plans for SOA composite applications.

Table 10–4 Configuration Plan Management Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_attachPlan	Attach the configuration plan file to the SOA composite application JAR file.	Offline
sca_extractPlan	Extract a configuration plan packaged with the JAR file for editing.	Offline

Table 10–4 (Cont.) Configuration Plan Management Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_generatePlan	Generate a configuration plan for editing.	Offline
sca_validatePlan	Validate the configuration plan.	Offline

10.4.1 sca_attachPlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

10.4.1.1 Description

Attaches the configuration plan file to the SOA composite application file. If a plan already exists in the file, it is overwritten with this new plan.

10.4.1.2 Syntax

```
sca_attachPlan(sar, configPlan, overwrite, verbose)
```

Argument	Definition
sar	Absolute path of the SAR file.
configPlan	Absolute path of the configuration plan file.
overwrite	Optional. Indicates whether to overwrite an existing configuration plan in the SAR file. ■ `false` (default): Does not overwrite the plan. ■ `true`: Overwrites the plan.
verbose	Optional. Indicates whether to print more information about the configuration plan attachment. ■ `true` (default): Prints more information. ■ `false`: Does not print more information.

10.4.1.3 Examples

The following example attaches the `configplan.xml` configuration plan file to the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_attachPlan("/tmp/sca_HelloWorld_rev1.0.jar",
"/tmp/configplan.xml")
```

The following example overwrites the existing configuration plan with `configplan.xml` file in the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_attachPlan("/tmp/sca_HelloWorld_rev1.0.jar",
"/tmp/configplan.xml", overwrite=true)
```

10.4.2 sca_extractPlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

10.4.2.1 Description

Extracts a configuration plan packaged with the SOA composite application file for editing, This is an optional step. If no plan exists, this is the same as creating a new file with `sca_generatePlan`.

10.4.2.2 Syntax

```
sca_extractPlan(sar, configPlan, overwrite, verbose)
```

Argument	Definition
sar	Absolute path of a SAR file.
configPlan	Absolute path of a configuration plan file to which to be extracted.
overwrite	Optional. Indicates whether to overwrite an existing configuration plan file in the SAR file. ■ `false` (default): Does not overwrite the plan. ■ `true`: Overwrites the plan.
verbose	Optional. Indicates whether to print more information about configuration plan extraction. ■ `true` (default): Prints more information. ■ `false`: Does not print more information.

10.4.2.3 Example

The following example extracts the `configplan.xml` file for editing from the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_extractPlan("/tmp/sca_HelloWorld_rev1.0.jar",
"/tmp/configplan.xml")
```

The following example extracts the `configplan.xml` file for editing from the `HelloWorld` application. This command also overwrites the existing plan.

```
wls:/mydomain/ServerConfig> sca_extractPlan("/tmp/sca_HelloWorld_rev1.0.jar",
"/tmp/configplan.xml", overwrite=true)
```

10.4.3 sca_generatePlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

10.4.3.1 Description

Generates a configuration plan for editing.

10.4.3.2 Syntax

```
sca_generatePlan(configPlan, sar, composite, overwrite, verbose)
```

Argument	Definition
configPlan	Absolute path of the configuration plan file to be generated.
sar	Absolute path of the SAR file.
composite	Absolute path of the `composite.xml` file in the expanded (unzipped) SAR directory.

Argument	Definition
overwrite	Optional. Indicates whether to overwrite an existing configuration plan file: ■ `false` (default): Does not overwrite the plan. ■ `true`: Overwrites the plan.
verbose	Indicates whether to print more information about plan generation: ■ `true` (default): Prints more information. ■ `false`: Does not print more information.

10.4.3.3 Examples

The following example generates the `myplan.xml` configuration plan file for the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_generatePlan("/tmp/myplan.xml",
sar="/tmp/sca_HelloWorld_rev1.0.jar")
```

The following example generates the `myplan2.xml` configuration plan file for the `HelloWorld` application. The `myplan2.xml` file overwrites the existing plan.

```
wls:/mydomain/ServerConfig> sca_generatePlan("/tmp/myplan2.xml",
composite="/tmp/HelloWorld_rev1.0/composite.xml", overwrite=true)
```

10.4.4 sca_validatePlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

10.4.4.1 Description

Validates the configuration plan. This command identifies all search and replacement changes to be made during deployment. Use this option for debugging only.

10.4.4.2 Syntax

```
sca_validatePlan(reportFile, configPlan, sar, composite, overwrite, verbose)
```

Argument	Definition
reportFile	Absolute path of the report file to be generated. Validation results are written to this file.
configPlan	Absolute path of the configuration plan file.
sar	Optional. The absolute path of the SAR file.
composite	Optional. The absolute path of the `composite.xml` file in the expanded (unzipped) SAR directory.
overwrite	Optional. Indicates whether to overwrite an existing configuration plan file: ■ `false` (default): Does not overwrite the plan. ■ `true`: Overwrites the plan.
verbose	Optional. Indicates whether to print more information about configuration plan validation. ■ `true` (default): Prints more information. ■ `false`: Does not print more information.

10.4.4.3 Examples

The following example validates the `configplan.xml` configuration plan file for the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_validatePlan("/tmp/myreport.xml",
"/tmp/configplan.xml", sar="/tmp/sca_HelloWorld_rev1.0.jar")
```

The following example validates the `configplan.xml` configuration plan file for the `HelloWorld` application. The `configplan.xml` plan overwrites the existing plan.

```
wls:/mydomain/ServerConfig> sca_validatePlan("/tmp/myreport.xml",
"/tmp/configplan.xml",composite="/tmp/HelloWorld_rev1.0/composite.xml",
overwrite=true)
```

10.5 Task Validation Commands

Use the task validation command, listed in Table 10–5, to validate human workflow tasks.

Table 10–5 Task Validation Command for WLST Configuration

Use this command...	To...	Use with WLST...
sca_validateTask	Validate a human workflow task.	Offline

10.5.1 sca_validateTask

Command Category: Task Validation Commands

Use with WLST: Offline

10.5.1.1 Description

Validates a human workflow task contained in the `.task` file that you created when designing a human task in the Human Task Editor.

10.5.1.2 Syntax

```
sca_validateTask(taskFile, outXml, displayLevel)
```

Argument	Definition
taskFile	Absolute path to the task definition file (`.task`).
outXml	Absolute path to an output XML file.
displayLevel	Optional. The level of information to display. The default value is `1`.

10.5.1.3 Example

The following example validates the `WFTaskDefinition.task` file of the human task.

```
wls:/mydomain/ServerConfig> sca_validateTask("/tmp/WFTaskDefinition.task",
"/tmp/out.xml", displayLevel=2)
```

10.6 SOA Composite Application Compilation Commands

Use the compilation commands, listed in Table 10–6, to compile SOA composite applications.

Table 10–6 SOA Composite Application Compilation Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_setProp	Set JVM system properties.	Offline
sca_compile	Compile a SOA composite application.	Offline

10.6.1 sca_setProp

Command Category: Application Compilation Commands

Use with WLST: Offline

10.6.1.1 Description

Sets JVM system properties. This command can also set secure socket layer (SSL) system properties before using `sca_deployComposite` and `sca_undeployComposite` over SSL.

10.6.1.2 Syntax

```
sca_setProp(propName, propValue)
```

Argument	Definition
propName	Property name.
propValue	Property value.

10.6.1.3 Example

The following example sets the property name and property value.

```
wls:/mydomain/ServerConfig> sca_setProp("oracle.home",
"/scratch/myusername/beahome/AS11gR1SOA")
```

10.6.2 sca_compile

Command Category: Application Compilation Commands

Use with WLST: Offline

10.6.2.1 Description

Compiles a SOA composite application.

> **Note:** The `sca_compile` command requires the `oracle.home` property to find the `ant-sca-compile.xml` script. This must be set once. You can use the `scac_setProp` command or the `oracleHome` property to set a value.

10.6.2.2 Syntax

```
sca_compile(composite, outXml, error, appHome, displayLevel, oracleHome,
configDir)
```

Argument	Definition
composite	Absolute path of a composite file in the expanded (unzipped) SAR directory.
outXml	Optional. Absolute path of an output XML file.
error	Optional. Absolute path of an error file.
appHome	Optional. Absolute path of the application home directory. This property is required if you have shared data.
displayLevel	Optional. The level of information to display. The default value is 1.
oracleHome	Optional. The oracle.home property.

10.6.2.3 Examples

The following example compiles the FirstComposite application.

```
wls:/mydomain/ServerConfig> sca_compile("/tmp/FirstComposite_
rev1.0/composite.xml", displayLevel=2)
```

The following example compiles the FirstComposite application and captures details in the myout.xml file. The error.out file captures any errors.

```
wls:/mydomain/ServerConfig> sca_compile("/tmp/FirstComposite_
rev1.0/composite.xml", outXml="/tmp/myout.xml", error="error.out")
```

The following example compiles the FirstComposite application. The oracleHome property is set to find the ant-sca-compile.xml script.

```
wls:/mydomain/ServerConfig> sca_compile("/tmp/FirstComposite_
rev1.0/composite.xml", displayLevel=2,
oracleHome="/scratch/myusername/beahome/AS11gR1SOA")
```

10.7 SOA Composite Application Packaging Commands

Use the packaging command, listed in Table 10–7, to package SOA composite applications into a composite SAR file.

Table 10–7 SOA Composite Application Packaging Command for WLST Configuration

Use this command...	To...	Use with WLST...
sca_package	Package the SOA composite application files into a composite SAR file.	Offline

10.7.1 sca_package

Command Category: Application Packaging Commands

Use with WLST: Offline

10.7.1.1 Description

Packages the SOA composite application files into a composite SAR file. This command performs the following operations:

- Calls sca_compile to compile the composite artifacts in ${compositeDir}.
- Calls javac to compile any source code under ${compositeDir}/src.
- Replaces the revision in ${compositeDir}/composite.xml.

- Packages the artifacts to create `sca_${compositeName}_rev${revision}.jar` in `${compositeDir}/deploy`.

> **Note:** The `sca_package` command requires `oracle.home` to find the `ant-sca-package.xml` script. This must be set once. You can use the `scac_setProp` command or `oracleHome` property to set this property.

10.7.1.2 Syntax

```
sca_package(compositeDir, compositeName, revision, appHome, oracleHome,
configDir)
```

Argument	Definition
compositeDir	Absolute path of a directory that contains composite artifacts.
compositeName	Name of the composite.
revision	Revision ID of the composite.
appHome	Optional. Absolute path of the application home directory. This property is required if you have shared data.
oracleHome	Optional. The `oracle.home` property.

10.7.1.3 Examples

The following example packages the `OrderBookingComposite` application. The `appHome` property is set because this application uses shared data.

```
wls:/mydomain/ServerConfig> sca_package("/tmp/app_data/OrderBookingComposite",
"OrderBookingComposite", "1.0", appHome="/tmp/app_data")
```

The following example packages the `HelloSOAComposite` application.

```
wls:/mydomain/ServerConfig> sca_package
("/tmp/HelloSOAApplication/HelloSOAComposite", "HelloSOAComposite", "1.0")
```

The following example packages the `HelloSOAComposite` application. The `oracleHome` property is set to find the `ant-sca-compile.xml` script.

```
wls:/mydomain/ServerConfig> sca_package
("/tmp/HelloSOAApplication/HelloSOAComposite", "HelloSOAComposite", "1.0",
oracleHome="/scratch/myusername/beahome/AS11gR1SOA")
```

10.8 SOA Composite Application Test Commands

Use the SOA composite application test command, listed in Table 10–8, to test a SOA composite applications.

Table 10–8 SOA Composite Application Test Command for WLST Configuration

Use this command...	To...	Use with WLST...
sca_test	Test deployed SOA composite applications.	Offline

10.8.1 sca_test

Command Category: Application Test Commands

Use with WLST: Offline

10.8.1.1 Description

Tests deployed SOA composite applications prior to deployment in a production environment. You create suites of tests in Oracle JDeveloper. The `sca_test` command calls `ant-sca-test.xml`.

10.8.1.2 Syntax

```
sca_test('compositeName', 'revision', 'testsuiteName', 'jndiPropFile',
oracleHome='oracleHome', javaHome='javaHome')
```

Argument	Definition
compositeName	Name of the SOA composite application.
revision	Revision ID of the SOA composite application.
testsuiteName	Name of the test suite.
jndiPropFile	Absolute path to the JNDI property file.
oracleHome	Optional. The `oracle.home` system property.
javaHome	Optional. The `java.passed.home` system property.

10.8.1.3 Examples

The following example runs the `OrderBookingMainTestsuite` test suite.

```
wls:/mydomain/ServerConfig> sca_test('OrderBookingComposite', '1.0',
 'OrderBookingMainTestsuite', '/tmp/tmp-jndi.properties',
 oracleHome='/scratch/<user>/beahome/AS11gR1SOA/',
 javaHome='/scratch/<user>/beahome/jdk160_05')
```

11

Application Development Framework (ADF) Custom WLST Commands

The following sections describe the WLST custom commands and variables in detail. Topics include:

- Section 11.1, "Overview of WSLT Command Categories"
- Section 11.2, "Commands for ADF-based URL Connections"

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

11.1 Overview of WSLT Command Categories

WLST commands are divided into the following categories.

Table 11–1 WLST Command Categories

Command Category	Description
Section 11.2, "Commands for ADF-based URL Connections"	Navigate the hierarchy of configuration or runtime beans and control the prompt display.

11.2 Commands for ADF-based URL Connections

Use the commands in Table 11–1 to managing URL-based connections.

Table 11–2 Browse Commands for WLST Configuration

Use this command...	To...	Use with WLST...
adf_createFileUrlConnection	Create a new ADF File connection.	Online or Offline
adf_createHttpUrlConnection	Create a new ADF URL connection.	Online or Offline
adf_ setURLConnectionAttributes	Set or edit the attributes of a newly created or existing ADF connection.	Online or Offline
adf_ setURLConnectionAttributes	Create a new connection.	Online or Offline

11.2.1 adf_createFileUrlConnection

Use with WLST: Online or Offline

11.2.1.1 Description

Creates a new connection based on the `oracle.adf.model.connection.url.FileURLConnection` connection class.

11.2.1.2 Syntax

```
adf_createFileURLConnection(appName, name, URL)
```

Argument	Definition
appName	Application name for which the connection that will be created.
name	The name of the new connection.
URL	The URL associated with this connection.

11.2.1.3 Example

```
adf_createFileURLConnection('myapp','tempDir','/scratch/tmp')
```

11.2.2 adf_createHttpUrlConnection

Use with WLST: Online or Offline

11.2.2.1 Description

Use this command to create a new connection based on the `oracle.adf.model.connection.url.HttpURLConnection` connection type class.

11.2.2.2 Syntax

```
adf.createHttpURLConnection (appName, name, [URL], [authenticationType], [realm],
[user], [password]
```

Argument	Definition
appName	Application name for which the connection is to be created.
name	The name of the new connection.
url	(Optional) The URL associated with this connection.
authenticationType	(Optional) The default is basic.
realm	(Optional) If this connection deals with authentication, then this should be set. The default is basic.
user	(Optional)
password	(Optional)

11.2.2.3 Example

```
adf_createHttpURLConnection('myapp','cnn','http://www.cnn.com')
```

11.2.3 adf_setURLConnectionAttributes

Use with WLST: Online or Offline

11.2.3.1 Description

Use this command to set or edit the attributes of a newly created or existing ADF connection.

11.2.3.2 Syntax

```
adf_setURLConnectionAttributes(appname, connectionname, attributes)
```

Argument	Definition
appname	Application name for which the connection that will be created.
connectionname	The name of the new connection.
attributes	The array containing attributes to set in key/value pairs.

11.2.3.3 Example

```
adf_setURLConnectionAttributes
('myapp','cnn','ChallengeAuthenticationType:digest',
'AuthenticationRealm:XMLRealm'
```

11.2.4 adf_listUrlConnection

Use with WLST: Online or Offline

11.2.4.1 Description

Use this command to create a new connection based on the `oracle.adf.model.connection.url.FileURLConnection` connection class.

11.2.4.2 Syntax

```
adf_listURLConnection(appname, name, URL)
```

Argument	Definition
appname	Application name for which the connection will be created.
name	The name of the new connection.
URL	The URL associated with this connection.

11.2.4.3 Example

```
adf_createFileURLConnection ('myapp','tempDir','/scratch/tmp')
```

12

Portal Custom WLST Commands

Portal custom WLST commands are extensions to the WLST commands and are specific to Oracle Portal. Table 12–1 lists the Portal custom WLST command categories.

For additional information about administration and configuration of Portal, see the *Oracle Portal Configuration Guide*.

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Table 12–1 Portal WLST Command Categories

Command category	Description
Section 12.1, "Database Access Descriptor Commands"	Create, edit, or delete a general DAD or Portal DAD.
Section 12.2, "Configuration Commands"	The Configuration commands: ■ List and update the WebCache configuration and Oracle Internet Directory data ■ Configure the Portal cache, Portal Page Engine, and Portal mid-tier ■ List Portal site configuration.

12.1 Database Access Descriptor Commands

A Database Access Descriptor (DAD) is a set of values that specify how an application connects to an Oracle database to fulfill an HTTP request. The information in the DAD includes the user name (which also specifies the schema and the privileges), password, connect string, and globalization support language of the database.

There are two types of DADs: general DAD and portal DAD. An Oracle Portal middle tier uses a Portal DAD to access the Oracle Metadata Repository. For information about general DADs, refer to the Oracle Fusion Middleware Administrator's Guide for Oracle HTTP Server.

Use the Database Access Descriptor commands listed in Table 12–2 to create, edit, or delete a Portal DAD from the WLST command-line scripting interface. Based on your actions, the `portal_dads.conf` file is updated.

Table 12–2 Database Access Descriptor Commands for Portal WLST Configuration

Use this command...	To...	Use with WLST...
listDads	List the parameters used by the Database Access Descriptors for configuration.	Online
createPortalDad	Create a Portal Database Access Descriptor.	Online
updatePortalDad	Update the attributes of a Portal Database Access Descriptor.	Online
deletePortalDad	Delete a Portal Database Access Descriptor.	Online

12.1.1 listDads

Command Category: Database Access Descriptor Commands

Use with WLST: Online

12.1.1.1 Description

Lists the parameters specified in all the Database Access Descriptors (both general DADs and Portal DADs).

12.1.1.2 Syntax

```
listDads ()
```

12.1.1.3 Example

The following example lists the various DADs in the domain.

```
listDads()
------------
/pls/portal1
Schema: h1user
Connect String: foo.oracle.com:1521:orcl
NLS Language: "AMERICAN_AMERICA.AL32UTF8"
```

12.1.2 createPortalDad

Command Category: Database Access Descriptor Commands

Use with WLST: Online

12.1.2.1 Description

Creates a Portal Database Access Descriptor.

12.1.2.2 Syntax

```
createPortalDad (name, schema, password, [connect_string], nls_language)
```

Argument	Definition
name	Name of the Database Access Descriptor.
schema	The Portal database account user name.
password	The Portal database account password.

Argument	Definition
connect_string	Optional. The connection string used to connect to a remote database. Connect string may be host name: port number: connect string. The connect string format may be ServiceNameFormat (host:port:database_service_name), SIDFormat (host:port:database_sid), or TNSFormat (TNS alias or the whole TNS entry).
nls_language	The globalization support language of the Portal database that is represented by this DAD. This setting overrides the NLS_LANG environment variable for a database session and defines some important globalization support properties of the response, including the response character set. Make sure that this language setting matches the NLS_LANG of the back-end database.

12.1.2.3 Example

The following example creates the portal1 Portal DAD based on the specified arguments.

```
createPortalDad(name='portal1',schema='schema',password='welcome1',connect_
string='foo.oracle.com:1521:orcl',nls_language='AMERICAN_AMERICA.AL32UTF8')
```

12.1.3 updatePortalDad

Command Category: Database Access Descriptor Commands

Use with WLST: Online

12.1.3.1 Description

Updates the attributes of the Portal Database Access Descriptor.

12.1.3.2 Syntax

```
updatePortalDad (name, [schema], [password], [connect_string], [nls_language])
```

Argument	Definition
name	Name of the Database Access Descriptor. This name cannot be changed during update.
schema	Optional. The Portal database account user name.
password	Optional. The Portal database account password.
connect_string	Optional. The connection string used to connect to a remote database. Connect string may be host name: port number: connect string. The connect string format may be ServiceNameFormat (host:port:database_service_name), SIDFormat (host:port:database_sid), or TNSFormat (TNS alias or the whole TNS entry).
nls_language	Optional. The globalization support language of the Portal database that is represented by this DAD. This setting overrides the NLS_LANG environment variable for a database session and defines some important Globalization Support properties of the response, including the response character set. Make sure that this language setting matches the NLS_LANG of the back-end database.

12.1.3.3 Example

The following example updates the portal1 Portal DAD based on the specified arguments.

```
updatePortalDad(name='portal1',schema='user1',password='welcome2',connect_
string='foo.oracle.com:1521:orcl',nls_language='AMERICAN_AMERICA.AL32UTF8')
```

12.1.4 deletePortalDad

Command Category: Database Access Descriptor Commands

Use with WLST: Online

12.1.4.1 Description

Deletes a Portal Database Access Descriptor.

12.1.4.2 Syntax

```
deletePortalDad(name)
```

Argument	Definition
name	Name of the Portal Database Access Descriptor.

12.1.4.3 Example

The following example deletes the portal1 Portal DAD entry from the portal_dads.conf file.

```
deletePortalDad(name='portal1')
```

12.2 Configuration Commands

Use the Configuration commands in Table 12–3 to view and configure Portal cache, WebCache, Oracle Internet Directory data and so on.

Table 12–3 Configuration Commands for the Portal WLST Configuration

Use this command...	To...	Use with WLST...
configurePortalCache	Update the attributes of the Portal cache.	Online
configurePortalPageEngine	Update the attributes of the Portal mid-tier.	Online
listPortalWebcacheConfigAttributes	List the attributes of WebCache configuration.	Online
listPortalSiteConfigAttributes	List the attributes of Portal site configuration.	Online
listPortalOIDConfigAttributes	List the attributes of Oracle Internet Directory configuration.	Online
setPortalWebcacheConfig	Update the attributes of the WebCache configuration.	Online
setPortalOIDConfig	Update the attributes of the Oracle Internet Directory configuration.	Online
setPortalMidtierConfig	Update the attributes of the Portal mid-tier configuration.	Online

12.2.1 configurePortalCache

Command Category: Configuration Commands

Use with WLST: Online

12.2.1.1 Description

Portal cache is a file system-based cache for Oracle Portal pages and portlets. Portal cache supports validation-based caching and expiry-based caching. Portal cache consists of both Portal content cache and session cache.

This command updates the attributes of the Portal cache. These configuration details are maintained in the <Middleware Home>/user_projects/domains/<DOMAIN_HOME>/servers/WLS_PORTAL/stage/portal/portal/configuration/portal_cache.conf file.

12.2.1.2 Syntax

```
configurePortalCache([enable], [directory], [total_size], [max_size],
[cleanup_time], [max_age])
```

Argument	Definition
enable	Optional. Enables (On) or disables (Off) portal content and session caching.
directory	Optional. The directory where cached content is stored. Make sure that this directory exists and has read-write access.
total_size	Optional. The total amount of disk space (in megabytes) that the Portal cache may use. The maximum value allowed is 4 GB.
max_size	Optional. The maximum size (in bytes) for all cached files. The maximum value allowed is 4 GB. Any dynamically generated content that exceeds this limit is not cached.
cleanup_time	Optional. The time at which to start the cleanup of the cache storage. Use the [Sunday-Saturday, Everyday, Everymonth][hh:mm] format to define the exact day and time in which cleanup should occur.
max_age	Optional. Maximum age of a single cached document. This setting ensures the cache system does not contain any old content. Old cache files are removed to make space for new cache files. The default is 30 days.

12.2.1.3 Example

The following example configures the Portal cache.

```
configurePortalCache(enable=true,directory='/scratch/user/installs/Inst_1
/cache/PortalComponent/portal',total_size=10101010,max_size=12300033,cleanup_
time='Everyday 11:00',max_age=20)
```

12.2.2 configurePortalPageEngine

Command Category: Configuration Commands

Use with WLST: Online

12.2.2.1 Description

The Oracle Fusion Middleware Portal architecture is designed around a three-tier architecture that allows any browser to connect to it. This flexible architecture allows each component (browser, Oracle HTTP Server listener, Oracle Database 11g, and Oracle Portal) to be upgraded individually as required.

A part of the Oracle Portal middle tier, the Parallel Page Engine (PPE) is a servlet that runs under Oracle Containers for J2EE and services page requests. The PPE reads page metadata, calls providers for portlet content, accepts provider responses, and assembles the requested page in the specified page layout.

This command updates the properties in the `appConfig.xml` file, the configuration file that is used by the Portal mid-tier repository servlet. This configuration file is located in the $MWHOME/user_projects/domains/AllClassicDomain/servers/WLS_PORTAL/stage/portal/portal/configuration/ directory.

12.2.2.2 Syntax

```
configurePortalPageEngine([encrypt_key], [resource_url_key], [use_port], [use_
scheme], [x509certfile])
```

Argument	Definition
`encrypt_key`	Optional. Specifies the HMCA key to obscure the headers used for caching using WebCache. This allows for a more secure cache key, and makes retrieving a cached object by unwanted requests more difficult.
`resource_url_key`	Optional. This key, used by the PPE servlet, calculates checksums for URLs that are requested by WSRP and JPDK resource proxying. For WSRP resource proxying to work, the key must be set to an alpha-numeric value of 10 characters or more. In addition, for JPDK proxying, a JNDI environment variable, also called resourceUrlKey, must be set for the provider.
`use_port`	Optional. Overrides the port used when the PPE makes requests to the portal. The default, if not specified, is to always use the page request port. Note that if you set useScheme, you must also set the usePort argument. This may be used for other reasons, but mostly it is used when SSL is running between the browser and the PPE but not between the PPE and Portal. In this case, the non-SSL port for loop back requests will be different from the SSL port used by the browser.
`use_scheme`	Optional. Overrides the scheme (HTTP or HTTPS) used when the PPE makes requests to the Portal. The default, if not specified, is to always use the page request scheme. Note that if you set useScheme, you must also set the usePort argument.
`x509certfile`	Optional. Specifies a file containing a list of certificates to be implicitly trusted by HTTPClient. These certificates are added as trust points to all connections made by HTTPClient using SSL.

12.2.2.3 Example

The following example updates the Portal page engine based on the specified arguments.

```
configurePortalPageEngine(encrypt_key='encryption key',resource_url_
key='foo.oracle.com',use_port=9999,use_scheme='page_engine_1',
x509certfile='file')
```

12.2.3 listPortalWebcacheConfigAttributes

Command Category: Configuration Commands

Use with WLST: Online

12.2.3.1 Description

Lists the attributes of WebCache configuration used by the Portal repository.

12.2.3.2 Syntax

```
listPortalWebcacheConfigAttributes ([dad_name])
```

Argument	Definition
dad_name	Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.

12.2.3.3 Example

The following example lists the WebCache configuration used by the Portal repository. The WebCache host name to which the invalidation messages are sent, the invalidation user name, password and the invalidation port to which the invalidation messages are sent are listed.

```
listPortalWebcacheConfigAttributes(dad_name='portal1')
listPortalWebcacheConfigAttributes('portal1')
---------------
WebCacheConfig
---------------
WebCache Host: foo.oracle.com
WebCache Invalidation Password: invalidator
WebCache Invalidation Port: 6523
WebCache Invalidation User: invalidator
```

12.2.4 listPortalSiteConfigAttributes

Command Category: Configuration Commands

Use with WLST: Online

12.2.4.1 Description

Lists the attributes of the Portal site configuration.

12.2.4.2 Syntax

```
listPortalSiteConfigAttributes ([dad_name])
```

Argument	Definition
dad_name	Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.

12.2.4.3 Example

The following example lists the Portal site configuration. Site protocol can be true or false. HTTP is the protocol when site protocol is false and HTTPS is the protocol when the site protocol is true. The site host name and port number are also listed.

```
listPortalSiteConfigAttributes(dad_name='portal1')
```

```
listPortalSiteConfigAttributes('portal1')

---------------
SiteConfig
---------------
Site Protocol: false
Site Host: foo.oracle.com
Site Port: 8090
```

12.2.5 listPortalOIDConfigAttributes

Command Category: Configuration Commands

Use with WLST: Online

12.2.5.1 Description

Lists the attributes of the Oracle Internet Directory configuration.

12.2.5.2 Syntax

```
listPortalOIDConfigAttributes ([dad_name])
```

Argument	Definition
dad_name	Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.

12.2.5.3 Example

The following example lists the Oracle Internet Directory data, which includes the Oracle Internet Directory host name and port number.

```
listPortalOIDConfigAttributes(dad_name='portal1')
listPortalOIDConfigAttributes('portal1')

---------------
OidConfig
---------------
OID Port: 13060
OID Host: foo.oracle.com
```

12.2.6 setPortalWebcacheConfig

Command Category: Configuration Commands

Use with WLST: Online

12.2.6.1 Description

WebCache offers caching, page assembly, and compression features. Oracle WebCache accelerates the delivery of both static and dynamic Web content, and provides load balancing and failover features for Oracle Fusion Middleware.

This command updates the WebCache configuration.

12.2.6.2 Syntax

```
setPortalWebcacheConfig([dad_name], [host], [inv_port], [inv_user],
[inv_passwd])
```

Argument	Definition
dad_name	Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.
host	Optional. The name of the WebCache host to which invalidation messages are sent.
inv_port	Optional. The WebCache port number to which invalidation messages are sent.
inv_user	Optional. The user name used for sending the invalidation messages.
inv_password	Optional. WebCache invalidation password.

12.2.6.3 Example

The following example updates the WebCache configuration based on the specified values.

```
setPortalWebcacheConfig(dad_name='portal1',host='foo.oracle.com',
inv_port= '6523',inv_user= 'invalidator',inv_passwd=' invalidator')
```

12.2.7 setPortalOIDConfig

Command Category: Configuration Commands

Use with WLST: Online

12.2.7.1 Description

Updates the attributes of the Oracle Internet Directory configuration.

12.2.7.2 Syntax

```
setPortalOIDConfig ([dad_name], [host], [port], [protocol], [admin_user],
[admin_passwd])
```

Argument	Definition
dad_name	Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.
host	Optional. Oracle Internet Directory host name.
port	Optional. Oracle Internet Directory port number.
protocol	Optional. Oracle Internet Directory protocol.
admin_user	Optional. Oracle Internet Directory administrator's name.
admin_passwd	Optional. Oracle Internet Directory administrator's password.

12.2.7.3 Example

The following example updates the OID configuration based on the specified values.

```
setPortalOIDConfig(dad_name='portal1',
host='foo.oracle.com',port='13060',protocol=false,
admin_user='cn=orcladmin',admin_passwd='oracle1')
```

12.2.8 setPortalMidtierConfig

Command Category: Configuration Commands

Use with WLST: Online

12.2.8.1 Description

Updates the Portal repository with the latest Portal mid-tier configuration.

12.2.8.2 Syntax

```
setPortalMidtierConfig([dad_name], [ohs_host], [ohs_port], [ohs_protocol],
[webcache_host], [webcache_inv_user], [webcache_inv_port],
[webcache_inv_passwd])
```

Argument	Definition
dad_name	Optional. Name of the Database Access Descriptor. Default DAD name is 'portal'.
ohs_host	Optional. Oracle HTTP Server host name.
ohs_port	Optional. Oracle HTTP Server port number.
ohs_protocol	Optional. Oracle HTTP Server protocol.
webcache_host	Optional. The name of the WebCache host to which invalidation messages are sent.
webcache_inv_user	Optional. The WebCache user name used for sending the invalidation messages.
webcache_inv_port	Optional. The WebCache port number to which invalidation messages are sent.
webcache_inv_passwd	Optional. WebCache invalidation password.

12.2.8.3 Example

The following example updates the Portal mid-tier configuration based on the specified values.

```
setPortalMidtierConfig(dad_name='portal1',ohs_host='foo.oracle.com',
ohs_port='8090',ohs_protocol=false,webcache_host='foo.oracle.com',
webcache_inv_user= 'invalidator',webcache_inv_port='6523',
webcache_inv_passwd='invalidator')
```

13

Java Required Files Custom WLST Commands

Java Required Files (JRF) consists of those components not included in the WebLogic Server installation that provide common functionality for Oracle business applications and application frameworks.

It consists of a number of independently developed libraries and applications that are deployed into a common location. The following components are considered part of Java Required Files: Oracle Application Development Framework, Oracle Fusion Middleware Audit Framework, Dynamic Monitoring Service, Fabric Common, HTTP Client, Infrastructure Security, Java Object Cache, JMX Framework, JPS, logging, MDS, OJSP.Next, Oracle Web Services, Oracle Web Services Manager, Oracle TopLink, UCP, XDK.

13.1 Java Required Files Commands

Use the commands in Table 13–1 to configure a Managed Server or cluster with Java Required Files (JRF) applications and services or to copy the applications and services from one Managed Server or cluster and apply them to another Managed Server or cluster.

In the Use with WLST column, online means the command can only be used when connected to a running server. Offline means the command can only be used when not connected to a running server. Online or offline means the command can be used in both situations.

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Table 13–1 DMS Commands

Use this command...	To...	Use with WLST...
Section 13.1.1, "applyJRF"	Configures a Managed Server or cluster with Java Required Files applications and services.	Online or Offline
Section 13.1.2, "cloneDeployments"	Copies the applications and services from Managed Server or cluster and applies them to another Managed Server or cluster.	Online or Offline

13.1.1 applyJRF

Use with WLST: Online or Offline

13.1.1.1 Description

Configures a Managed Server or cluster with Java Required Files (JRF). Managed Servers that are added by product templates during the template extension process do not need to be explicitly configured with JRF using this command.

Use the applyJRF command when additional Managed Servers or clusters are added to a domain after it is initially extended with a product template. The applyJRF command is required any time you add a Managed Server to a JRF-only domain, or if you add a Managed Server that has been configured for JRF to a domain that contains other Oracle products.

13.1.1.2 Syntax

```
applyJRF(target, [domainDir], [shouldUpdateDomain])
```

Argument	Definition
`target`	The name of the Managed Server or cluster to be configured with JRF applications and services. A value of an asterisk (*) for the target indicates that all clusters and standalone Managed Servers should be configured with JRF.
`domainDir`	The absolute path of the WebLogic Server domain.
`shouldUpdateDomain`	An optional boolean flag that controls how domain updates are carried out. When you set it to true (the default), the function implicitly invokes the following offline commands: readDomain() and updateDomain(), or the online commands: edit(), startEdit(), save(), and activate(). When you set it to false, you must call WLST commands to update the domain.

13.1.1.3 Example

The following example configures the Managed Server server1 with JRF:

```
wls:/offline> applyJRF('server1', '/my_path/user_templates/domains/my_domain')
```

13.1.2 cloneDeployments

Use with WLST: Online or Offline

13.1.2.1 Description

Replicates all deployments targeted to a particular Managed Server or cluster on a second Managed Server or cluster. This command is provided as a convenience to configure a new Managed Server or cluster so that it has the same deployments as a pre-existing Managed Server or cluster.

The cloneDeployments command does not create new Managed Servers, and it does not copy properties other than deployment information to the target Managed Server.

13.1.2.2 Syntax

```
cloneDeployments(domain, source, target, [shouldUpdateDomain])
```

Argument	Definition
domain	The absolute path of the WebLogic Server domain. Ignored if the domain has been read, or if connected in online mode.
source	The name of the Managed Server or cluster from which you want to clone deployments. This must be the name of a valid Managed Server or cluster.
target	The target Managed Server or cluster that will receive the source server's applications and services. The target Managed Server must already exist.
shouldUpdateDomain	An optional boolean flag that controls how domain updates are carried out. When you set it to true (the default), the function implicitly invokes the following offline commands: readDomain() and updateDomain(), or online commands: edit(), startEdit(), save(), and activate(). When you set it to false, you must call WLST commands to update the domain.

13.1.2.3 Example

The following example replicates the deployments from sourceServer to destinationServer:

```
wls:/offline> cloneDeployments( '/my_path/user_templates/domains/my_domain',
 'sourceServer','destinationServer', 'false')
```