Oracle® Fusion Middleware

WebLogic Scripting Tool Command Reference

11*g* Release 1 (10.3.6)

E13813-11

November 2011

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

ORACLE

Oracle Fusion Middleware WebLogic Scripting Tool Command Reference, 11*g* Release 1 (10.3.6)

E13813-11

Contents

4 Infrastructure Security Custom WLST Commands

7 Logging Custom WLST Commands

8 Metadata Services (MDS) Custom WLST Commands

9 Oracle SOA Suite Custom WLST Commands

10 WebCenter Portal Custom WLST Commands

11 Application Development Framework (ADF) Custom WLST Commands

12 Portal Custom WLST Commands

13 Java Required Files Custom WLST Commands

14 Web Services Custom WLST Commands

15 Diagnostic Framework Custom WLST Commands

16 Information Rights Management Custom WLST Commands

19 Oracle WebCenter Content Custom WLST Commands

20 Enterprise Scheduling Service (ESS) Custom WLST Commands

Preface

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

Documentation Accessibility

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

Access to Oracle Support

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

Conventions

The following text conventions are used in this document:

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

1 Introduction and Roadmap

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

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

1.1 Document Scope and Audience

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

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

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

1.2 Guide to This Document

This document is organized as follows:

- This chapter, "Introduction and Roadmap," introduces the organization of this guide and lists related documentation.
- Chapter 2, "WebLogic Server WLST Online and Offline Command Reference," summarizes WebLogic Server WLST commands alphabetically and by online/offline usage.
- Chapter 3, "WLST Command and Variable Reference," provides detailed descriptions for each of the WebLogic Server WLST commands and variables.
- 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, "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 6, "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 7, "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 8, "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 9, "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 10, "WebCenter Portal 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 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.
- Chapter 14, "Web Services Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Web Services component.
- Chapter 15, "Diagnostic Framework Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Diagnostic Framework component.
- Chapter 16, "Information Rights Management Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Fusion Middleware Information Rights Management component.
- Chapter 17, "Oracle WebCenter: Imaging Custom WLST Commands," provides detailed descriptions of the custom WLST commands that can be used to manage the Oracle Imaging and Process Management component.
- Chapter 18, "Oracle Business Process Management Custom WLST Commands," provides detailed descriptions of the custom WLST commands for Oracle Business Process Management.
- Chapter 19, "Oracle WebCenter Content Custom WLST Commands," provides detailed descriptions of the custom WLST commands for Oracle WebCenter Content.

- Chapter 20, "Enterprise Scheduling Service (ESS) Custom WLST Commands," provides detailed descriptions of the custom WLST commands for Oracle Enterprise Scheduling Service (ESS).

1.3 Related Documentation

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

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

- "Using Ant Tasks to Configure and Use a WebLogic Server Domain" in *Developing Applications for Oracle WebLogic Server*, describes using WebLogic Ant tasks for starting and stopping WebLogic Server instances and configuring WebLogic domains.
- "Deployment Tools" in *Deploying Applications to Oracle WebLogic Server* describes several tools that WebLogic Server provides for deploying applications and stand-alone modules.
- *Administration Console Online Help* describes a Web-based graphical user interface for managing and monitoring WebLogic domains.
- *Creating WebLogic Domains Using the Configuration Wizard* describes using a graphical user interface to create a WebLogic domain or extend an existing one.
- *Creating Templates and Domains Using the Pack and Unpack Commands* describes commands that recreate existing WebLogic domains quickly and easily.
- *Developing Custom Management Utilities With JMX for Oracle WebLogic Server* describes using Java Management Extensions (JMX) APIs to monitor and modify WebLogic Server resources.
- *SNMP Management Guide for Oracle WebLogic Server* describes using Simple Network Management Protocol (SNMP) to monitor WebLogic domains.
- *Oracle Fusion Middleware Administrator's Guide* describes how to manage Oracle Fusion Middleware, including how to start and stop Oracle Fusion Middleware, how to configure and reconfigure components, and how to back up and recover.

1.4 New and Changed WLST Features in This Release

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

2

WebLogic Server WLST Online and Offline Command Reference

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 WebLogic 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 WebLogic domain.	Offline
closeTemplate	Close the current domain template.	Offline
configToScript	Convert an existing server configuration (`config` directory) to an executable WLST script.	Online or Offline
connect	Connect WLST to a WebLogic Server instance.	Online or Offline
create	Create a configuration bean of the specified type for the current bean.	Online or Offline
currentTree	Return the current location in the hierarchy.	Online
custom	Navigate to the root of custom MBeans that are registered in the Runtime MBean Server.	Online
delete	Delete an instance of a configuration bean of the specified type for the current configuration bean.	Online or Offline
deploy	Deploy an application to a WebLogic Server instance.	Online
disconnect	Disconnect WLST from a WebLogic Server instance.	Online
distributeApplication	Copy the deployment bundle to the specified targets.	Online
domainConfig	Navigate to the last MBean to which you navigated in the domain configuration hierarchy or to the root of the hierarchy, `DomainMBean`.	Online
domainCustom	Navigate to the tree of custom MBeans that are registered in the Domain Runtime MBean Server.	Online
domainRuntime	Navigate to the last MBean to which you navigated in the domain runtime hierarchy or to the root of the hierarchy, `DomainRuntimeMBean`.	Online
dumpStack	Display stack trace from the last exception that occurred while performing a WLST action, and reset the stack trace.	Online or Offline

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

This command...	Enables you to...	Use with WLST...
dumpVariables	Display all variables used by WLST, including their name and value.	Online or Offline
edit	Navigate to the last MBean to which you navigated in the configuration edit MBean hierarchy or to the root of the hierarchy, `DomainMBean`.	Online
encrypt	Encrypt the specified string.	Online
exit	Exit WLST from the user session and close the scripting shell.	Online or Offline
exportDiagnosticData	Execute a query against the specified log file.	Offline
exportDiagnosticDataFromServer	Executes a query on the server side and retrieves the exported WebLogic Diagnostic Framework (WLDF) data.	Online
find	Find MBeans and attributes in the current hierarchy.	Online
get	Return the value of the specified attribute.	Online or Offline
getActivationTask	Return the latest `ActivationTask` MBean on which a user can get status.	Online
getAvailableCapturedImages	Returns a list of the previously captured diagnostic images.	Online
getConfigManager	Return the latest `ConfigurationManagerBean` MBean which manages the change process.	Online
getMBean	Return the MBean by browsing to the specified path.	Online
getMBI	Return the `MBeanInfo` for the specified `MBeanType` or the `cmo` variable.	Online
getPath	Return the MBean path for the specified MBean instance.	Online
getWLDM	Return the WebLogic `DeploymentManager` object.	Online
invoke	Invoke a management operation on the current configuration bean.	Online
isRestartRequired	Determine whether a server restart is required.	Online
jndi	Navigates to the JNDI tree for the server to which WLST is currently connected.	Online
listApplications	List all applications that are currently deployed in the domain.	Online
listChildTypes	List all the children MBeans that can be created or deleted for the `cmo`.	Online
loadApplication	Load an application and deployment plan into memory.	Online or Offline
loadDB	Load SQL files into a database.	Offline
loadProperties	Load property values from a file.	Online and Offline
lookup	Look up the specified MBean.	Online

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

This command...	Enables you to...	Use with WLST...
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
nmDisconnect	Disconnect WLST from a Node Manager session.	Online or Offline
nmEnroll	Enroll the machine on which WLST is currently running.	Online
nmGenBootStartupProps	Generates the Node Manager property files, `boot.properties` and `startup.properties`, for the specified server.	Online
nmKill	Kill the specified server instance that was started with Node Manager.	Online or Offline
nmLog	Return the Node Manager log.	Online or Offline
nmServerLog	Return the server output log of the server that was started with Node Manager.	Online or Offline
nmServerStatus	Return the status of the server that was started with Node Manager.	Online or Offline
nmStart	Start a server in the current domain using Node Manager.	Online or Offline
nmVersion	Return the Node Manager server version.	Online or Offline
prompt	Toggle the display of path information at the prompt.	Online or Offline
pwd	Display the current location in the configuration or runtime bean hierarchy.	Online or Offline
readDomain	Open an existing WebLogic domain for updating.	Offline
readTemplate	Open an existing domain template for WebLogic domain creation.	Offline
redeploy	Reload classes and redeploy a previously deployed application.	Online

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

This command...	Enables you to...	Use with WLST...
redirect	Redirect WLST output to the specified filename.	Online or Offline
removeListener	Remove a listener that was previously defined.	Online
resume	Resume a server instance that is suspended or in `ADMIN` state.	Online
save	Save the edits that have been made but have not yet been saved.	Online
saveDiagnosticImageCaptureFile	Downloads the specified diagnostic image capture.	Online
saveDiagnosticImageCaptureEntryFile	Downloads a specific entry from the diagnostic image capture.	Online
serverRuntime	Navigate to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, `ServerRuntimeMBean`.	Online
set	Set the specified attribute value for the current configuration bean.	Online or Offline
setOption	Set options related to a WebLogic domain creation or update	Offline
showChanges	Show the changes made by the current user during the current edit session.	Online
showListeners	Show all listeners that are currently defined.	Online
shutdown	Gracefully shut down a running server instance or cluster.	Online
start	Start a Managed Server instance or a cluster using Node Manager.	Online
startApplication	Start an application, making it available to users.	Online
startEdit	Start a configuration edit session on behalf of the currently connected user.	Online
startNodeManager	Start Node Manager at default port (5556).	Online or Offline
startRecording	Record all user interactions with WLST; useful for capturing commands to replay.	Online or Offline
startServer	Start the Administration Server.	Online or Offline
state	Returns a map of servers or clusters and their state using Node Manager.	Online
stopApplication	Stop an application, making it un available to users.	Online
stopEdit	Stop the current edit session, release the edit lock, and discard unsaved changes.	Online
stopNodeManager	Stop Node Manager.	Online or Offline

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

This command...	Enables you to...	Use with WLST...
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
writeDomain	Write the domain configuration information to the specified directory.	Offline
writeIniFile	Convert WLST definitions and method declarations to a Python (`.py`) file.	Online or Offline
writeTemplate	Writes the domain configuration information to the specified domain template.	Offline

2.2 WebLogic Server WLST Online Command Summary

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

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

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

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

This command...	Enables you to...
getAvailableCapturedImages	Returns a list of the previously captured diagnostic images.
getConfigManager	Return the latest `ConfigurationManagerBean` MBean which manages the change process.
getMBean	Return the MBean by browsing to the specified path.
getMBI	Return the `MBeanInfo` for the specified `MBeanType` or the `cmo` variable.
getPath	Return the MBean path for the specified MBean instance.
getWLDM	Return the WebLogic `DeploymentManager` object.
invoke	Invoke a management operation on the current configuration bean.
isRestartRequired	Determine whether a server restart is required.
jndi	Navigates to the JNDI tree for the server to which WLST is currently connected.
listApplications	List all applications that are currently deployed in the domain.
listChildTypes	List all the children MBeans that can be created or deleted for the `cmo`.
loadApplication	Load an application and deployment plan into memory.
loadProperties	Load property values from a file.
lookup	Look up the specified MBean.
ls	List all child beans and/or attributes for the current configuration or runtime bean.
man	Display help from `MBeanInfo` for the current MBean or its specified attribute.
migrate	Migrate services to a target server within a cluster.
nm	Determine whether WLST is connected to Node Manager.
nmConnect	Connect WLST to Node Manager to establish a session.
nmDisconnect	Disconnect WLST from a Node Manager session.
nmEnroll	Enroll the machine on which WLST is currently running.
nmGenBootStartupProps	Generates the Node Manager property files, `boot.properties` and `startup.properties`, for the specified server.
nmKill	Kill the specified server instance that was started with Node Manager.
nmLog	Return the Node Manager log.
nmServerLog	Return the server output log of the server that was started with Node Manager.
nmServerStatus	Return the status of the server that was started with Node Manager.
nmStart	Start a server in the current domain using Node Manager.
nmVersion	Return the Node Manager server version.
prompt	Toggle the display of path information at the prompt.

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

This command...	Enables you to...
pwd	Display the current location in the configuration or runtime bean hierarchy.
redeploy	Reload classes and redeploy a previously deployed application.
redirect	Redirect WLST output to the specified filename.
removeListener	Remove a listener that was previously defined.
resume	Resume a server instance that is suspended or in `ADMIN` state.
save	Save the edits that have been made but have not yet been saved.
saveDiagnosticImageCaptureFile	Downloads the specified diagnostic image capture.
saveDiagnosticImageCaptureEntryFile	Downloads a specific entry from the diagnostic image capture.
serverConfig	Navigate to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, `DomainMBean`.
serverRuntime	Navigate to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, `ServerRuntimeMBean`.
set	Set the specified attribute value for the current configuration bean.
showChanges	Show the changes made by the current user during the current edit session.
showListeners	Show all listeners that are currently defined.
shutdown	Gracefully shut down a running server instance or cluster.
start	Start a Managed Server instance or a cluster using Node Manager.
startApplication	Start an application, making it available to users.
startEdit	Start a configuration edit session on behalf of the currently connected user.
startNodeManager	Start Node Manager at default port (5556).
startRecording	Record all user interactions with WLST; useful for capturing commands to replay.
startServer	Start the Administration Server.
state	Returns a map of servers or clusters and their state using Node Manager
stopApplication	Stop an application, making it un available to users.
stopEdit	Stop the current edit session, release the edit lock, and discard unsaved changes.
stopNodeManager	Stop Node Manager.
stopRedirect	Stop the redirection of WLST output to a file.
storeUserConfig	Create a user configuration file and an associated key file.
suspend	Suspend a running server.
threadDump	Display a thread dump for the specified server.
undeploy	Undeploy an application from the specified servers.

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

This command...	Enables you to...
undo	Revert all unsaved or unactivated edits.
updateApplication	Update an application configuration using a new deployment plan.
validate	Validate the changes that have been made but have not yet been saved.
viewMBean	Display information about an MBean, such as the attribute names and values, and operations.
writeIniFile	Convert WLST definitions and method declarations to a Python (`.py`) file.

2.3 WebLogic Server WLST Offline Command Summary

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

Table 2–3 WebLogic Server WLST Offline Command Summary

This command...	Enables you to...
addHelpCommand	Adds new command help for a command to an existing command group. Once added to the group, the command (along with a brief description) is displayed in the command list for the group when you enter the `help('commandGroup')` command.
addHelpCommandGroup	Adds a new help command group to those shown by the WLST `help()` command, and specifies the resource bundle in which the help information is defined for the group.
addTemplate	Extend the current domain using an application or service extension template.
assign	Assign resources to one or more destinations.
cd	Navigate the hierarchy of configuration or runtime beans.
closeDomain	Close the current domain.
closeTemplate	Close the current domain template.
configToScript	Convert an existing server configuration (`config` directory) to an executable WLST script.
connect	Connect WLST to a WebLogic Server instance.
create	Create a configuration bean of the specified type for the current bean.
delete	Delete an instance of a configuration bean of the specified type for the current configuration bean.
dumpStack	Display stack trace from the last exception that occurred while performing a WLST action, and reset the stack trace.
dumpVariables	Display all variables used by WLST, including their name and value.
exit	Exit WLST from the interactive session and close the scripting shell.
exportDiagnosticData	Execute a query against the specified log file.
get	Return the value of the specified attribute.

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

This command...	Enables you to...
loadApplication	Load an application and deployment plan into memory.
loadDB	Load SQL files into a database.
loadProperties	Load property values from a file.
ls	List all child beans and/or attributes for the current configuration or runtime bean.
nmConnect	Connect WLST to Node Manager to establish a session.
prompt	Toggle the display of path information at the prompt.
pwd	Display the current location in the configuration or runtime bean hierarchy.
readDomain	Open an existing WebLogic domain for updating.
readTemplate	Open an existing domain template for domain creation.
redirect	Redirect WLST output to the specified filename.
set	Set the specified attribute value for the current configuration bean.
setOption	Set options related to a WebLogic domain creation or update.
startNodeManager	Start Node Manager at default port (5556).
startRecording	Record all user interactions with WLST; useful for capturing commands to replay.
startServer	Start the Administration Server.
stopNodeManager	Stop Node Manager.
stopRedirect	Stop the redirection of WLST output to a file.
threadDump	Display a thread dump for the specified server.
unassign	Unassign applications or services from one or more destinations.
updateDomain	Update and save the current domain.
writeDomain	Write the domain configuration information to the specified directory.
writeIniFile	Convert WLST definitions and method declarations to a Python (`.py`) file.
writeTemplate	Writes the domain configuration information to the specified domain template.

3

WLST Command and Variable Reference

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

3.1 Overview of WLST Command Categories

Note: It is recommended that you review "Syntax for WLST Commands" in *Oracle WebLogic Scripting Tool* for command syntax requirements.

WLST commands are divided into the following categories.

Table 3–1 WLST Command Categories

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

Table 3–1 (Cont.) WLST Command Categories

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

3.2 Browse Commands

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

Table 3–2 Browse Commands for WLST Configuration

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

3.2.1 cd

Command Category: Browse Commands

Use with WLST: Online or Offline

3.2.1.1 Description

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

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

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

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

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

3.2.1.2 Syntax

```
cd(mbeanName)
```

Argument	Definition
mbeanName	Path to the bean in the namespace.

3.2.1.3 Examples

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

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

3.2.2 currentTree

Command Category: Browse Commands

Use with WLST: Online

3.2.2.1 Description

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

3.2.2.2 Syntax

```
currentTree()
```

3.2.2.3 Example

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

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

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

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

3.2.3 prompt

Command Category: Browse Commands

Use with WLST: Online or Offline

3.2.3.1 Description

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

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

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

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

3.2.3.2 Syntax

```
prompt(myPrompt)
```

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

3.2.3.3 Examples

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

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

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

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

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

3.2.4 pwd

Command Category: Browse Commands

Use with WLST: Online or Offline

3.2.4.1 Description

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

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

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

3.2.4.2 Syntax

```
pwd()
```

3.2.4.3 Example

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

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

3.3 Control Commands

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

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

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

Table 3–3 Control Commands for WLST Configuration

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

3.3.1 addTemplate

Command Category: Control Commands

Use with WLST: Offline

3.3.1.1 Description

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

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

3.3.1.2 Syntax

```
addTemplate(templateFileName)
```

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

3.3.1.3 Example

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

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

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

3.3.3 closeTemplate

Command Category: Control Commands

Use with WLST: Offline

3.3.3.1 Description

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

3.3.3.2 Syntax

```
closeTemplate()
```

3.3.3.3 Example

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

```
wls:/offline> readTemplate('c:/Oracle/Middleware/wlserver_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 in development mode, WebLogic Server encrypts the credentials that were used to create the server and stores them in a `boot.properties` file. When you create an Administration Server in production mode, no `boot.properties` file is created. If your production domain does not contain a `boot.properties` file, you can create one manually; see "Creating a Boot Identify File for an Administration Server" in *Managing Server Startup and Shutdown for Oracle WebLogic Server*.

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

Please note:

- If you run the `connect` command in a script without specifying the username and password or user configuration file and key file, a `WSLTException` occurs. In interactive mode, you are prompted for the username and password.
- Oracle strongly recommends that you connect WLST to the server through the SSL port or administration port. If you do not, the following warning message is displayed:

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

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

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

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

- If you are connecting to a WebLogic Server instance via HTTP, ensure that the `TunnelingEnabled` attribute is set to `true` for the WebLogic Server instance. For more information, see "TunnelingEnabled" in *Oracle WebLogic Server MBean Reference*.
- When trying to connect to the WebLogic Server Administration Server from WLST using localhost as the host name, the following message may be displayed if the listen-address attribute of the Administration Server has been restricted to certain IP addresses:

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

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

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

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

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

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

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

3.3.4.2 Syntax

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

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

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

3.3.4.3 Examples

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

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

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

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

wls:/mydomain/serverConfig>
```

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

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

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

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

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

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

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

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

3.3.5 createDomain

Command Category: Control Commands

Use with WLST: Offline

3.3.5.1 Description

Creates a WebLogic domain using the specified template.

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

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

3.3.5.2 Syntax

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

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

3.3.5.3 Example

The following example creates a new WebLogic domain using the Avitek MedRec template and sets the default username to `weblogic` and the password to `welome1`. The domain is saved to the following directory: `c:/Oracle/Middleware/wlserver_10.3/user_projects/domains/medrec`.

```
wls:/offline> createDomain('c:/Oracle/Middleware/wlserver_10.3/common
/templates/domains/wls_medrec.jar','c:/Oracle/Middleware/user_
projects/domains/medrec',
'weblogic', 'welcome1')
```

3.3.6 disconnect

Command Category: Control Commands

Use with WLST: Online

3.3.6.1 Description

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

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

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

3.3.6.2 Syntax

```
disconnect(force)
```

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

3.3.6.3 Example

The following example disconnects from a running server:

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

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

3.3.7 exit

Command Category: Control Commands

Use with WLST: Online or Offline

3.3.7.1 Description

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

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

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

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

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

3.3.7.2 Syntax

```
exit([defaultAnswer], [exitcode])
```

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

3.3.7.3 Example

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

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

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

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

3.3.8 readDomain

Command Category: Control Commands

Use with WLST: Offline

3.3.8.1 Description

Opens an existing WebLogic domain for updating.

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

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

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

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

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

3.3.8.2 Syntax

```
readDomain(domainDirName)
```

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

3.3.8.3 Example

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

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

3.3.9 readTemplate

Command Category: Control Commands

Use with WLST: Offline

3.3.9.1 Description

Opens an existing domain template for domain creation.

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

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

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

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

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

3.3.9.2 Syntax

```
readTemplate(templateFileName)
```

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

3.3.9.3 Example

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

```
wls:/offline> readTemplate('c:/Oracle/Middleware/wlserver_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 WebLogic domain. The domain continues to be editable after you update and save it.

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

3.3.10.2 Syntax

```
updateDomain()
```

3.3.10.3 Example

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

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

3.3.11 writeDomain

Command Category: Control Commands

Use with WLST: Offline

3.3.11.1 Description

Writes the domain configuration information to the specified directory.

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

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

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

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

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

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

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

3.3.11.2 Syntax

```
writeDomain(domainDir)
```

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

3.3.11.3 Example

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

```
wls:/offline> readTemplate('c:/Oracle/Middleware/wlserver_10.3/common/templates
/domains/wls.jar')
...
wls:/offline/base_domain> writeDomain('c:/Oracle/Middleware/user_
projects/domains/base_domain')
```

3.3.12 writeTemplate

Command Category: Control Commands

Use with WLST: Offline

3.3.12.1 Description

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

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

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

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

3.3.12.2 Syntax

```
writeTemplate(templateName)
```

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

3.3.12.3 Example

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

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

3.4 Customization Commands

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

Table 3–4 Customization Commands for WLST Configuration

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

3.4.1 addHelpCommandGroup

Command Category: Customization Commands

Use with WLST: Online or Offline

3.4.1.1 Description

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

3.4.1.2 Syntax

```
addHelpCommandGroup(commandGroup, resourceBundleName)
```

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

3.4.1.3 Examples

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

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

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

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

3.4.2 addHelpCommand

Command Category: Customization Commands

Use with WLST: Online or Offline

3.4.2.1 Description

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

3.4.2.2 Syntax

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

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

3.4.2.3 Example

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

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

3.5 Deployment Commands

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

- Deploy, undeploy, and redeploy applications and standalone modules to a WebLogic Server instance.
- Update an existing deployment plan.
- Interrogate the WebLogic Deployment Manager object.
- Start and stop a deployed application.

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

Table 3–5 Deployment Commands for WLST Configuration

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

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

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

3.5.1 deploy

Command Category: Deployment Commands

Use with WLST: Online

3.5.1.1 Description

Deploys an application to a WebLogic Server instance.

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

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

3.5.1.2 Syntax

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

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

Argument	Definition
options	Optional. Comma-separated list of deployment options, specified as name-value pairs. Valid options include: ■ **altDD**—Location of the alternate application deployment descriptor on the Administration Server. ■ **altWlsDD**—Location of the alternate WebLogic application deployment descriptor on the Administration Server. ■ **archiveVersion**—Archive version number. ■ **block**—Boolean value specifying whether WLST should block user interaction until the command completes. This option defaults to `true`. If set to `false`, WLST returns control to the user after issuing the command; you can query the `WLSTProgress` object to determine the status of the command. If you are importing WLST as a Jython module, as described in "Importing WLST as a Jython Module" in *Oracle WebLogic Scripting Tool*, *`block`* is always set to `true`. ■ **clusterDeploymentTimeout**—Time, in milliseconds, granted for a cluster deployment task on this application. ■ **createPlan**—Boolean value indicating that user would like to create a default plan. This option defaults to `false`. ■ **defaultSubmoduleTargets**—Boolean value indicating that targeting for qualifying JMS submodules should be derived by the system, see "Using Sub-Module Targeting with JMS Application Modules" in *Deploying Applications to Oracle WebLogic Server*. Default value is true. ■ **deploymentPrincipalName**—String value specifying the principal for deploying the file or archive during server starts (static deployment; it does not effect the current deployment task). Make sure the user exists. This option adds `<deployment-principal-name>` to the `<app-deployment>` element in the `config.xml` file. ■ **forceUndeployTimeout**—Force undeployment timeout value. ■ **gracefulIgnoreSessions**—Boolean value specifying whether the graceful production to admin mode operation should ignore pending HTTP sessions. This option defaults to `false` and only applies if `gracefulProductionToAdmin` is set to `true`. ■ **gracefulProductionToAdmin**—Boolean value specifying whether the production to Admin mode operation should be graceful. This option defaults to `false`. ■ **libImplVersion**—Implementation version of the library, if it is not present in the manifest. ■ **libraryModule**—Boolean value specifying whether the module is a library module. This option defaults to `false`.

Argument	Definition
options *(Continued)*	■ **libSpecVersion**—Specification version of the library, if it is not present in the manifest. ■ **planVersion**—Plan version number. ■ **remote**—Boolean value specifying whether the operation will be remote from the file system that contains the source. Use this option when you are on a different machine from the Administration Server and the deployment files are already at the specified location where the Administration Server is located. This option defaults to false. ■ **retireGracefully**—Retirement policy to gracefully retire an application only after it has completed all in-flight work. This policy is only meaningful for stop and redeploy operations and is mutually exclusive to the retire timeout policy. ■ **retireTimeout**—Time (in seconds) WLST waits before retiring an application that has been replaced with a newer version. This option default to `-1`, which specifies graceful timeout. ■ **securityModel**—Security model. Valid values include: `DDOnly`, `CustomRoles`, `CustomRolesAndPolicies`, and `Advanced`. ■ **securityValidationEnabled**—Boolean value specifying whether security validation is enabled. ■ **subModuleTargets**—Submodule level targets for JMS modules. For example, `submod@mod-jms.xml@target | submoduleName@target`. ■ **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 WebLogic Scripting Tool*.

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

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

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

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

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

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

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

3.5.2 distributeApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.2.1 Description

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

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

3.5.2.2 Syntax

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

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

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

3.5.2.3 Example

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

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

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

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

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

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

3.5.3 getWLDM

Command Category: Deployment Commands

Use with WLST: Online

3.5.3.1 Description

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

3.5.3.2 Syntax

```
getWLDM()
```

3.5.3.3 Example

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

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

3.5.4 listApplications

Command Category: Deployment Commands

Use with WLST: Online

3.5.4.1 Description

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

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

3.5.4.2 Syntax

```
listApplications()
```

3.5.4.3 Example

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

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

3.5.5 loadApplication

Command Category: Deployment Commands

Use with WLST: Online and Offline

3.5.5.1 Description

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

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

3.5.5.2 Syntax

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

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

3.5.5.3 Example

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

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

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

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

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

3.5.6 redeploy

Command Category: Deployment Commands

Use with WLST: Online

3.5.6.1 Description

Reloads classes and redeploys a previously deployed application.

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

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

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

3.5.6.2 Syntax

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

Argument	Definition
appName	Name of the application to be redeployed.

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

3.5.6.3 Example

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

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

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

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

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

3.5.7 startApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.7.1 Description

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

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

3.5.7.2 Syntax

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

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

3.5.7.3 Example

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

```
wls:/mydomain/serverConfig/Servers> progress=startApplication('BigApp',
stageMode='NOSTAGE', 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 WebLogic Scripting Tool*.

3.5.8 stopApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.8.1 Description

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

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

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

3.5.8.2 Syntax

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

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

3.5.8.3 Example

The following example stops the `BigApp` application.

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

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

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

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

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

3.5.9 undeploy

Command Category: Deployment Commands

Use with WLST: Online

3.5.9.1 Description

Undeploys an application from the specified servers.

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

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

3.5.9.2 Syntax

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

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

3.5.9.3 Example

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

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

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

3.5.10 updateApplication

Command Category: Deployment Commands

Use with WLST: Online

3.5.10.1 Description

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

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

3.5.10.2 Syntax

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

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

3.5.10.3 Example

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

```
wls:/offline> progress=updateApplication('BigApp',
'c:/myapps/BigApp/newPlan/plan.xml', stageMode='STAGE', 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 WebLogic Scripting Tool*.

3.6 Diagnostics Commands

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

Table 3–6 Diagnostic Command for WLST Configuration

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

3.6.1 exportDiagnosticData

Command Category: Diagnostics Commands

Use with WLST: Offline

3.6.1.1 Description

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

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

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

3.6.1.2 Syntax

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

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

3.6.1.3 Example

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

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

Exporting diagnostic data to export.xml
<Aug 2, 2005 6:58:21 PM EDT> <Info> <Store> <BEA-280050> <Persistent store
 "WLS_DIAGNOSTICS" opened: directory="c:\Oracle\Middleware
\wlserver_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 *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*.

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

3.6.2.2 Syntax

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

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

3.6.2.3 Example

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

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

3.6.3 getAvailableCapturedImages

Command Category: Diagnostics Commands

Use with WLST: Online

3.6.3.1 Description

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

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

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

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

3.6.3.2 Syntax

```
getAvailableCapturedImages()
```

3.6.3.3 Example

The following example returns an array of strings named `images`, which contains a list of the diagnostic image capture files available in the image destination directory, and prints the entries contained in the diagnostic image named `diagnostic_image_myserver_2009_06_15_14_58_36.zip`.

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

3.6.4 saveDiagnosticImageCaptureFile

Command Category: Diagnostics Commands

Use with WLST: Online

3.6.4.1 Description

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

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

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

3.6.4.2 Syntax

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

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

3.6.4.3 Example

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

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

3.6.5 saveDiagnosticImageCaptureEntryFile

Command Category: Diagnostics Commands

Use with WLST: Online

3.6.5.1 Description

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

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

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

3.6.5.2 Syntax

```
saveDiagnosticImageCaptureEntryFile(imageName, imageEntryName, [outputFile])
```

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

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

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

3.6.5.3 Example

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

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

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

3.7 Editing Commands

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

Note: To edit configuration beans, you must be connected to an Administration Server, and you must navigate to the edit tree and start an edit session, as described in Section 3.11.5, "edit" and Section 3.7.17, "startEdit", respectively.

If you connect to a Managed Server, WLST functionality is limited to browsing the configuration bean hierarchy. While you cannot use WLST to change the values of MBeans on Managed Servers, it is possible to use the Management APIs to do so. Oracle recommends that you change only the values of configuration MBeans on the Administration Server. Changing the values of MBeans on Managed Servers can lead to an inconsistent domain configuration.

For more information about editing configuration beans, see "Using WLST Online to Update an Existing Domain" in *Oracle WebLogic Scripting Tool*.

Table 3–7 Editing Commands for WLST Configuration

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

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

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

3.7.1 activate

Command Category: Editing Commands

Use with WLST: Online

3.7.1.1 Description

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

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

3.7.1.2 Syntax

```
activate([timeout], [block])
```

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

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

3.7.1.3 Example

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

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

3.7.2 assign

Command Category: Editing Commands

Use with WLST: Offline

3.7.2.1 Description

Assigns resources to one or more destinations.

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

3.7.2.2 Syntax

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

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

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

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

- When assigning **application deployments**, set the values as follows:
  - *sourceType*: `AppDeployment`
  - *destinationType*: `Target`
- When assigning **libraries**, set the values as follows:
  - *sourceType*: `Library`
  - *destinationType*: `Target`
- When assigning **services**, set the values as follows:
  - *sourceType*: Name of the specific server, such as `JDBCSystemResource`
  - *destinationType*: `Target`
- When assigning **servers** to **clusters**, set the values as follows:
  - *sourceType*: `Server`
  - *destinationType*: `Cluster`
- When assigning **subdeployments**, set the values as follows:
  - *sourceType*: *service*`.SubDeployment`, where *service* specifies the parent of the `SubDeployment`, such as `JMSSystemResource.SubDeployment`; you can also specify nested subdeployments (such as `AppDeployment.SubDeployment.SubDeployment`)
  - *destinationType*: `Target`
- When assigning **security types**, set the values as follows:
  - *sourceType*: Name of the security type, such as `User`
  - *destinationType*: Name of the destination security type, such as `Group`

3.7.2.3 Example

The following examples:

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

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

- Assign all servers to the cluster `myCluster`.

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

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

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

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

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

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

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

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

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

3.7.3 cancelEdit

Command Category: Editing Commands

Use with WLST: Online

3.7.3.1 Description

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

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

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

3.7.3.2 Syntax

```
cancelEdit([defaultAnswer])
```

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

3.7.3.3 Example

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

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

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

3.7.4 create

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.4.1 Description

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

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

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

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

- You must be connected to an Administration Server. You cannot use the `create` command for runtime MBeans or when WLST is connected to a Managed Server instance.
- You must navigate to the edit configuration MBean hierarchy using the `edit` command before issuing this command. See Section 3.11.5, "edit".
- You can use the create command to create a WebLogic Server configuration MBean that is a child of the current MBean type.

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

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

For more information about:

- Creating MBeans, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities with JMX*.
- Examples of creating specific types of MBean resources, for example, a JMS or JDBC system resource, refer to the WLST sample scripts installed with your product, as described in "WLST Sample Scripts" in *Oracle WebLogic Scripting Tool*.
- MBeans, their child types, attributes, and operations, see *Oracle WebLogic Server MBean Reference*.

3.7.4.2 Syntax

```
create(name, childMBeanType, [baseProviderType])
```

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

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

3.7.4.3 Example

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

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

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

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

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

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

3.7.5 delete

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.5.1 Description

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

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

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

3.7.5.2 Syntax

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

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

3.7.5.3 Example

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

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

3.7.6 encrypt

Command Category: Editing Commands

Use with WLST: Online

3.7.6.1 Description

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

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

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

3.7.6.2 Syntax

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

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

3.7.6.3 Example

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

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

3.7.7 get

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.7.1 Description

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

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

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

```
cmo.getListenPort()
```

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

3.7.7.2 Syntax

```
get(attrName)
```

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

3.7.7.3 Example

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

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

Alternatively, you can use the `cmo` variable:

```
cmo.getAdministrationPort()
```

3.7.8 getActivationTask

Command Category: Editing Commands

Use with WLST: Online

3.7.8.1 Description

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

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

3.7.8.2 Syntax

```
getActivationTask()
```

3.7.8.3 Example

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

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

3.7.9 invoke

Command Category: Editing Commands

Use with WLST: Online

3.7.9.1 Description

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

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

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

3.7.9.2 Syntax

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

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

3.7.9.3 Example

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

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

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

3.7.10 isRestartRequired

Command Category: Editing Commands

Use with WLST: Online

3.7.10.1 Description

Determines whether a server restart is required.

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

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

3.7.10.2 Syntax

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

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

3.7.10.3 Example

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

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

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

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

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

3.7.11 loadDB

Command Category: Editing Commands

Use with WLST: Offline

3.7.11.1 Description

Loads SQL files into a database.

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

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

- The appropriate database is running.
- SQL files exist for the specified database and version.

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

The command fails if the above conditions are not met.

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

3.7.11.2 Syntax

```
loadDB(dbVersion, datasourceName, dbCategory)
```

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

3.7.11.3 Example

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

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

3.7.12 loadProperties

Command Category: Editing Commands

Use with WLST: Online and Offline

3.7.12.1 Description

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

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

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

3.7.12.2 Syntax

```
loadProperties(fileName)
```

Argument	Definition
fileName	Properties file pathname.

3.7.12.3 Example

This example gets and sets the properties file values.

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

3.7.13 save

Command Category: Editing Commands

Use with WLST: Online

3.7.13.1 Description

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

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

3.7.13.2 Syntax

```
save()
```

3.7.13.3 Example

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

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

3.7.14 set

Command Category: Editing Commands

Use with WLST: Online or Offline

3.7.14.1 Description

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

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

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

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

- You must be in an edit session to use this command. See Section 3.7.17, "startEdit".
- You cannot use this command when WLST is connected to a Managed Server.
- As an alternative to this command, you can use the `cmo` variable with the following syntax:

  `cmo.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 WebLogic Scripting Tool*.

3.7.14.2 Syntax

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

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

3.7.14.3 Example

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

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

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

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

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

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

3.7.15 setOption

Command Category: Editing Commands

Use with WLST: Offline

3.7.15.1 Description

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

3.7.15.2 Syntax

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

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

3.7.15.3 Example

The following example sets the `CreateStartMenu` option to `false`:

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

3.7.16 showChanges

Command Category: Editing Commands

Use with WLST: Online

3.7.16.1 Description

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

3.7.16.2 Syntax

```
showChanges([onlyInMemory])
```

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

3.7.16.3 Example

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

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

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

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

3.7.17 startEdit

Command Category: Editing Commands

Use with WLST: Online

3.7.17.1 Description

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

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

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

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

3.7.17.2 Syntax

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

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

3.7.17.3 Example

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

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

3.7.18 stopEdit

Command Category: Editing Commands

Use with WLST: Online

3.7.18.1 Description

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

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

3.7.18.2 Syntax

```
stopEdit([defaultAnswer])
```

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

3.7.18.3 Example

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

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

3.7.19 unassign

Command Category: Editing Commands

Use with WLST: Offline

3.7.19.1 Description

Unassign applications or resources from one or more destinations.

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

3.7.19.2 Syntax

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

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

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

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

  - *sourceType*: AppDeployment
  - *destinationType*: Target
- When unassigning **libraries**, set the values as follows:
  - *sourceType*: Library
  - *destinationType*: Target
- When unassigning **security types**, set the values as follows:
  - *sourceType*: Name of the security type, such as User
  - *destinationType*: Name of the destination security type, such as Group
- When unassigning **servers** from **clusters**, set the values as follows:
  - *sourceType*: Server
  - *destinationType*: Cluster
- When unassigning **services**, set the values as follows:
  - *sourceType*: Name of the specific server, such as JDBCSystemResource
  - *destinationType*: Target
- When unassigning **subdeployments**, set the values as follows:
  - *sourceType*: *service*.SubDeployment, where *service* specifies the parent of the SubDeployment, such as JMSSystemResource.SubDeployment; you can also specify nested subdeployments (such as AppDeployment.SubDeployment.SubDeployment)
  - *destinationType*: Target

3.7.19.3 Example

The following examples:

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

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

- Unassign all servers from the cluster myCluster.

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

- Unassign the user newUser from the group Monitors.

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

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

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

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

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

3.7.20 undo

Command Category: Editing Commands

Use with WLST: Online

3.7.20.1 Description

Reverts all unsaved or unactivated edits.

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

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

3.7.20.2 Syntax

```
undo([unactivatedChanges], [defaultAnswer])
```

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

3.7.20.3 Example

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

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

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

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

3.7.21 validate

Command Category: Editing Commands

Use with WLST: Online

3.7.21.1 Description

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

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

3.7.21.2 Syntax

```
validate()
```

3.7.21.3 Example

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

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

3.8 Information Commands

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

Table 3–8 Information Commands for WLST Configuration

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

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

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

3.8.1 addListener

Command Category: Information Commands

Use with WLST: Online

3.8.1.1 Description

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

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

3.8.1.2 Syntax

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

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

3.8.1.3 Example

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

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

3.8.2 configToScript

Command Category: Information Commands

Use with WLST: Online or Offline

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

The configToScript command creates the following files:

- A WLST script that contains the commands needed to recreate the configuration.
- A properties file that contains domain-specific values. You can update the values in this file to create new domains that are similar to the original configuration.
- A user configuration file and an associated key file to store encrypted attributes. The user configuration file contains the encrypted information. The key file contains a secret key that is used to encrypt and decrypt the encrypted information.

When you run the generated script:

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

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

3.8.2.1 Syntax

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

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

3.8.2.2 Example

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

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

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

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

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

3.8.3 dumpStack

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.3.1 Description

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

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

3.8.3.2 Syntax

```
dumpStack()
```

3.8.3.3 Example

This example displays the stack trace.

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

3.8.4 dumpVariables

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.4.1 Description

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

3.8.4.2 Syntax

```
dumpVariables()
```

3.8.4.3 Example

This example displays all the current variables and their values.

```
wls:/mydomain/serverConfig> dumpVariables()
adminHome   weblogic.rmi.internal.BasicRemoteRef - hostID:
   '-1 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 WebLogic domain. In the event of an error, the command returns a `WLSTException`.

3.8.6.2 Syntax

```
getConfigManager()
```

3.8.6.3 Example

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

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

3.8.7 getMBean

Command Category: Information Commands

Use with WLST: Online

3.8.7.1 Description

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

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

3.8.7.2 Syntax

```
getMBean(mbeanPath)
```

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

3.8.7.3 Example

The following example returns the MBean specified by the path.

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

3.8.8 getMBI

Command Category: Information Commands

Use with WLST: Online

3.8.8.1 Description

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

3.8.8.2 Syntax

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

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

3.8.8.3 Example

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

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

3.8.9 getPath

Command Category: Information Commands

Use with WLST: Online

3.8.9.1 Description

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

3.8.9.2 Syntax

```
getPath(mbean)
```

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

3.8.9.3 Example

The following example returns the MBean specified by the path.

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

3.8.10 listChildTypes

Command Category: Information Commands

Use with WLST: Online

3.8.10.1 Description

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

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

3.8.10.2 Syntax

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

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

3.8.10.3 Example

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

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

3.8.11 lookup

Command Category: Information Commands

Use with WLST: Online

3.8.11.1 Description

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

3.8.11.2 Syntax

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

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

3.8.11.3 Example

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

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

3.8.12 ls

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.12.1 Description

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

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

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

- The first column displays a set of codes that describe the listed item. See Table 3–9.
- The second column displays the item name.
- When the item is an attribute, the third column displays the attribute value. If an attribute is encrypted, the third column displays asterisks instead of the value. (See "Writing and Reading Encrypted Configuration Values" in *Oracle WebLogic Scripting Tool*.)
- When the item is an operation, the third column uses the following pattern to display the operation's return type and input parameters: *`returnType`*`:` *`parameterType`*`(`*`parameterName`*`)`

Table 3–9 ls Command Output Information

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

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

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

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

Note: As a performance optimization, when using WLST offline, WebLogic Server does not store most of its default values in the configuration files for the WebLogic domain. In some cases, this optimization prevents entire management objects from being displayed by WLST offline (because WebLogic Server has never written the corresponding XML elements to the domain configuration files). For example, if you never modify the default logging severity level for a WebLogic domain while the domain is active, WLST offline will not display the `Log` management object for the domain.

If you want to change the default value of attributes whose management object is not displayed by WLST offline, you must first use the `create` command to create the management object. Then you can `cd` to the management object and change the attribute value. See Section 3.7.4, "create".

3.8.12.2 Syntax

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

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

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

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

3.8.12.3 Example

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

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

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

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

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

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

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

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

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

3.8.13 man

Command Category: Information Commands

Use with WLST: Online

3.8.13.1 Description

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

3.8.13.2 Syntax

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

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

3.8.13.3 Example

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

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

3.8.14 redirect

Command Category: Information Commands

Use with WLST: Online

3.8.14.1 Description

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

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

3.8.14.2 Syntax

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

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

3.8.14.3 Example

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

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

3.8.15 removeListener

Command Category: Information Commands

Use with WLST: Online

3.8.15.1 Description

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

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

3.8.15.2 Syntax

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

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

3.8.15.3 Example

The following example removes the listener named `mylistener`.

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

3.8.16 showListeners

Command Category: Information Commands

Use with WLST: Online

3.8.16.1 Description

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

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

3.8.16.2 Syntax

```
showListeners()
```

3.8.16.3 Example

The following example shows all listeners that are currently defined.

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

3.8.17 startRecording

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.17.1 Description

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

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

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

3.8.17.2 Syntax

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

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

3.8.17.3 Example

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

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

3.8.18 state

Command Category: Information Commands

Use with WLST: Online

3.8.18.1 Description

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

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

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

3.8.18.2 Syntax

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

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

3.8.18.3 Example

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

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

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

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

States of the servers are
```

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

3.8.19 stopRecording

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.19.1 Description

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

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

3.8.19.2 Syntax

```
stopRecording()
```

3.8.19.3 Example

The following example stops recording WLST commands.

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

3.8.20 stopRedirect

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.20.1 Description

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

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

3.8.20.2 Syntax

```
stopRedirect()
```

3.8.20.3 Example

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

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

3.8.21 storeUserConfig

Command Category: Information Commands

Use with WLST: Online

3.8.21.1 Description

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

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

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

3.8.21.2 Syntax

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

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

3.8.21.3 Example

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

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

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

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

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

3.8.22 threadDump

Command Category: Information Commands

Use with WLST: Online or Offline

3.8.22.1 Description

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

3.8.22.2 Syntax

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

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

3.8.22.3 Example

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

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

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

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

3.8.23 viewMBean

Command Category: Information Commands

Use with WLST: Online

3.8.23.1 Description

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

3.8.23.2 Syntax

```
viewMBean(mbean)
```

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

3.8.23.3 Example

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

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

3.8.24 writeIniFile

Command Category: Information Commands

Use with WLST: Online

3.8.24.1 Description

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

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

3.8.24.2 Syntax

```
writeIniFile(filePath)
```

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

3.8.24.3 Example

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

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

3.9 Life Cycle Commands

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

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

Table 3–10 Life Cycle Commands for WLST Configuration

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

3.9.1 migrate

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.1.1 Description

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

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

3.9.1.2 Syntax

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

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

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

3.9.1.3 Example

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

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

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

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

3.9.2 resume

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.2.1 Description

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

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

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

3.9.2.2 Syntax

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

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

3.9.2.3 Example

The following example resumes a Managed Server instance.

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

3.9.3 shutdown

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.3.1 Description

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

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

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

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

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

3.9.3.2 Syntax

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

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

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

3.9.3.3 Example

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

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

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

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

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

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

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

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

States of the servers are
```

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

3.9.4 start

Command Category: Life Cycle Commands

Use with WLST: Online

3.9.4.1 Description

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

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

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

3.9.4.2 Syntax

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

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

3.9.4.3 Example

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

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

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

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

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

3.9.5 startServer

Command Category: Life Cycle Commands

Use with WLST: Online or Offline

3.9.5.1 Description

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

> **Note:** You can use `startServer` only to start a WebLogic Administration Server, by running WLST from the WL_HOME/common/bin directory. You cannot use `startServer` to start an integrated WebLogic Administration Server (that is, an Administration Server for a Fusion Middleware Suite product installed in an ORACLE_HOME directory).
>
> To start the Administration server for a Fusion Middleware Suite product other than WebLogic Server, use either of the following methods:
>
> - Execute the server startup script for the associated WebLogic domain.
> - Start the server using Node Manager. If you use this method, make sure that the `startScriptEnabled` property is set to `true` in Node Manager.

3.9.5.2 Syntax

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

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

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

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

3.9.6.2 Syntax

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

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

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

3.9.6.3 Example

The following example suspends a Managed Server instance:

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

3.10 Node Manager Commands

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

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

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

Table 3–11 Node Manager Commands for WLST Configuration

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

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

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

3.10.1 nm

Command Category: Node Manager Commands

Use with WLST: Online or Offline

3.10.1.1 Description

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

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

3.10.1.2 Syntax

```
nm()
```

3.10.1.3 Example

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

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

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

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

3.10.2 nmConnect

Command Category: Node Manager Commands

Use with WLST: Online or Offline

3.10.2.1 Description

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

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

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

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

3.10.2.2 Syntax

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

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

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

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

3.10.2.3 Example

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

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

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

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

3.10.3 nmDisconnect

Command Category: Node Manager Commands

Use with WLST: Online or Offline

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

3.10.3.1 Description

Disconnects WLST from a Node Manager session.

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

3.10.3.2 Syntax

```
nmDisconnect()
```

3.10.3.3 Example

The following example disconnects WLST from a Node Manager session.

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

3.10.4 nmEnroll

Command Category: Node Manager Commands

Use with WLST: Online

3.10.4.1 Description

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

This command downloads the following files from the Administration Server:

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

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

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

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

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

3.10.4.2 Syntax

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

Argument	Definition
domainDir	Optional. Path of the domain directory to which you want to save the Node Manager secret file (`nm_password.properties`) and `SerializedSystemIni.dat` file. This argument defaults to the directory in which WLST was started.
nmHome	Optional. Path to the Node Manager home. The `nodemanager.domains` file, containing the domain information, is written to this directory. This argument defaults to *`WL_HOME`*`/common/nodemanager`, where *`WL_HOME`* refers to the top-level installation directory for WebLogic Server.

3.10.4.3 Example

The following example enrolls the current machine with Node Manager and saves the Node Manager secret file (`nm_password properties`) and `SerializedSystemIni.dat` file to `c:/Oracle/Middleware/mydomain/common/nodemanager/nm_password.properties`. The `nodemanager.domains` file is written to *`WL_HOME`*`/common/nodemanager` by default.

```
wls:/mydomain/serverConfig>
nmEnroll('c:/Oracle/Middleware/mydomain/common/nodemanager')
Enrolling this machine with the domain directory at
c:\Oracle\Middleware\mydomain\common\nodemanager....
Successfully enrolled this machine with the domain directory at
C:\Oracle\Middleware\mydomain\common\nodemanager
wls:/mydomain/serverConfig>
```

3.10.5 nmGenBootStartupProps

Command Category: Node Manager Commands

Use with WLST: Online

3.10.5.1 Description

Generates the Node Manager property files, `boot.properties` and `startup.properties`, for the specified server. The Node Manager property files are stored relative to the root directory of the specified server. The target root directory must be on the same machine on which you are running the command.

You must specify the name of a server; otherwise, the command will fail.

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

3.10.5.2 Syntax

```
nmGenBootStartupProps(serverName)
```

Argument	Definition
`serverName`	Name of the server for which Node Manager property files are generated.

3.10.5.3 Example

The following example generates `boot.properties` and `startup.properties` in the root directory of the specified server, `ms1`.

```
wls:/mydomain/serverConfig> nmGenBootStartupProps('ms1')
Successfully generated boot.properties at
c:\Oracle\Middleware\mydomain\servers\ms1\data\nodemanager\boot.properties
Successfully generated startup.properties at
c:\Oracle\Middleware\mydomain\servers\ms1\data\nodemanager\startup.properties
wls:/mydomain/serverConfig>
```

3.10.6 nmKill

Command Category: Node Manager Commands

Use with WLST: Online or Offline

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

3.10.6.1 Description

Kills the specified server instance that was started with Node Manager.

If you do not specify a server name using the *serverName* argument, the argument defaults to `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], [serverType])
```

Argument	Definition
serverName	Optional. Name of the server to be killed. This argument defaults to `myserver`.
serverType	Optional. The type of server to start. This argument defaults to WebLogic. Another valid option is Coherence.

3.10.6.3 Example

The following example kills the server named `oamserver`.

```
wls:/nm/oamdomain> nmKill('oamserver')
Killing server 'oamserver' ...
Server oamServer killed successfully.
wls:/nm/oamdomain>
```

3.10.7 nmLog

Command Category: Node Manager Commands

Use with WLST: Online or Offline

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

3.10.7.1 Description

Returns the Node Manager log.

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

3.10.7.2 Syntax

```
nmLog([writer])
```

Argument	Definition
writer	Optional. `java.io.Writer` object to which you want to stream the log output. This argument defaults to the WLST writer stream.

3.10.7.3 Example

The following example displays the Node Manager log.

```
wls:/nm/oamdomain> nmLog()
Successfully retrieved the Node Manager log and written.
wls:/nm/oamdomain>
```

3.10.8 nmServerLog

Command Category: Node Manager Commands

Use with WLST: Online or Offline

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

3.10.8.1 Description

Returns the server output log of the server that was started with Node Manager.

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

3.10.8.2 Syntax

```
nmServerLog([serverName], [writer], [serverType])
```

Argument	Definition
serverName	Optional. Name of the server for which you want to display the server output log. This argument defaults to `myserver`.
`writer`	Optional. `java.io.Writer` object to which you want to stream the log output. This argument defaults to the WLSTInterpreter standard out, if not specified.
serverType	Optional. The type of server to start. This argument defaults to WebLogic. Another valid option is Coherence.

3.10.8.3 Example

The following example displays the server output log for the `oamserver` server and writes the log output to `myWriter`.

```
wls:/nm/oamdomain> nmServerLog('oamserver',myWriter)
Successfully retrieved the server log and written.
wls:/nm/oamdomain>
```

3.10.9 nmServerStatus

Command Category: Node Manager Commands

Use with WLST: Online or Offline

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

3.10.9.1 Description

Returns the status of the server that was started with Node Manager.

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

3.10.9.2 Syntax

```
nmServerStatus([serverName], [serverType])
```

Argument	Definition
serverName	Optional. Name of the server for which you want to display the status. This argument defaults to `myserver`.
serverType	Optional. The type of server to start. This argument defaults to WebLogic. Another valid option is Coherence.

3.10.9.3 Example

The following example displays the status of the server named `oamserver`, which was started with Node Manager.

```
wls:/nm/oamdomain> nmServerStatus('oamserver')
RUNNING
wls:/nm/oamdomain>
```

3.10.10 nmStart

Command Category: Node Manager Commands

Use with WLST: Online or Offline

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

3.10.10.1 Description

Starts a server in the current WebLogic domain using Node Manager.

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

Note: `boot.properties` must exist in order to start a server with `nmStart`. If this is the first time you are starting a server, you must manually create it in order to use `nmStart`.

Alternatively, you can use the `nmStartprops` argument to provide user credentials (after connecting to Node Manager):

```
prps = makePropertiesObject("username=weblogic, password=welcome1")
nmStart("AdminServer",props=prps)
```

3.10.10.2 Syntax

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

Argument	Definition
serverName	Optional. Name of the server to be started.
domainDir	Optional. Domain directory of the server to be started. This argument defaults to the directory from which you started WLST.
props	Optional. System properties to apply to the new server.
writer	Optional. `java.io.Writer` object to which the server output is written. This argument defaults to the WLST writer.
serverType	Optional. The type of server to start. This argument defaults to WebLogic. Another valid option is Coherence.

3.10.10.3 Example

The following example starts the `managed1` server in the current WebLogic domain using Node Manager.

```
wls:/nm/mydomain> nmStart("managed1")
Starting server managed1 ...
Server managed1 started successfully
wls:/nm/mydomain>
```

The following example starts the Administration Server in the specified WebLogic domain using Node Manager. In this example, the `prps` variable stores the system property settings and is passed to the command using the `props` argument.

```
wls:/nm/mydomain> prps = makePropertiesObject("weblogic.ListenPort=8001")
wls:/nm/mydomain> nmStart("AdminServer",props=prps)
Starting server AdminServer...
Server AdminServer started successfully
wls:/nm/mydomain>
```

3.10.11 nmVersion

Command Category: Node Manager Commands

Use with WLST: Online or Offline

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

3.10.11.1 Description

Returns the Node Manager version.

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

3.10.11.2 Syntax

```
nmVersion()
```

3.10.11.3 Example

The following example displays the Node Manager version.

```
wls:/nm/oamdomain> nmVersion()
The Node Manager version that you are currently connected to is 9.0.0.0
wls:/nm/oamdomain>
```

3.10.12 startNodeManager

Command Category: Node Manager Commands

Use with WLST: Online or Offline

3.10.12.1 Description

Starts Node Manager on the same computer that is running WLST.

Notes: The WebLogic Server custom installation process optionally installs and starts Node Manager as a Windows service on Windows systems. For more information, see "About Installing Node Manager as a Windows Service" in the *Installation Guide for Oracle WebLogic Server*. In this case, you do not need to start the Node Manager manually.

In production environments, Oracle recommends that you do *not* use the `startNodeManager` command to start Node Manager. The recommended approach is to install Node Manager as a service or daemon, or to use the startNodeManager script (`startNodeManager.sh` or `startNodeManger.cmd`).

If Node Manager is already running when you invoke the `startNodeManager` command, the following message is displayed:

```
A Node Manager has already been started.
Cannot start another Node Manager process via WLST
```

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

3.10.12.2 Syntax

```
startNodeManager([verbose], [nmProperties])
```

Argument	Definition
verbose	Optional. Boolean value specifying whether WLST starts Node Manager in verbose mode. This argument defaults to `false`, disabling verbose mode.
nmProperties	Optional. Comma-separated list of Node Manager properties, specified as name-value pairs. Node Manager properties include, but are not limited to, the following: `NodeManagerHome`, `ListenAddress`, `ListenPort`, and `PropertiesFile`.

3.10.12.3 Example

The following example displays the Node Manager server version.

```
wls:/mydomain/serverConfig> startNodeManager(verbose='true',
NodeManagerHome='c:/Oracle/Middleware/wlserver_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:\Oracle\Middleware\wlserver_
10.3\common\nodemanager.
wls:/mydomain/serverConfig>
```

3.10.13 stopNodeManager

Command Category: Node Manager Commands

Use with WLST: Online or Offline

3.10.13.1 Description

Stops the Node Manager process.

> **Note:** In order to stop the Node Manager process, you must have either started Node Manager with `startNodeManager`, or Node Manager must have been started with the property `QuitEnabled=true`. You can configure this property in `$WLS_HOME/common/nodemanager.properties`. This allows you to connect to the Node Manager to shut it down.

If the Node Manager is not running when you invoke the `stopNodeManager` command, the following message is displayed:

```
Cannot stop the Node Manager unless you are connected to it.
```

3.10.13.2 Syntax

```
stopNodeManager()
```

3.10.13.3 Example

The following example stops the Node Manager process for the `base_domain` domain.

```
wls:/nm/base_domain> stopNodeManager()
Stopped Node Manager Process successfully
wls:/offline>
```

3.11 Tree Commands

Use the WLST tree commands, listed in Table 3–12, to navigate among MBean hierarchies.

Table 3–12 Tree Commands for WLST Configuration

Use this command...	To...	Use with WLST...
custom	Navigate to the root of custom MBeans that are registered in the server.	Online
domainConfig	Navigate to the last MBean to which you navigated in the domain configuration hierarchy or to the root of the hierarchy, `DomainMBean`.	Online
domainCustom	Navigate to the root of custom MBeans that are registered in the Domain Runtime MBean Server	Online
domainRuntime	Navigate to the last MBean to which you navigated in the domain runtime hierarchy or to the root of the hierarchy, `DomainRuntimeMBean`.	Online
edit	Navigate to the last MBean to which you navigated in the edit configuration MBean hierarchy or to the root of the hierarchy, `DomainMBean`.	Online
jndi	Navigates to the JNDI tree for the server to which WLST is currently connected.	Online
serverConfig	Navigate to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, `DomainMBean`.	Online
serverRuntime	Navigate to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, `ServerRuntimeMBean`.	Online

3.11.1 custom

Command Category: Tree Commands

Use with WLST: Online

3.11.1.1 Description

Navigates to the root of custom MBeans that are registered in the Runtime MBean Server. WLST navigates, interrogates, and edits custom MBeans as it does domain MBeans; however, custom MBeans cannot use the `cmo` variable because a stub is not available.

Note: When navigating to the `custom` tree, WLST queries all MBeans in the compatibility MBean server, the runtime MBean server, and potentially the JVM platform MBean server to locate the custom MBeans. Depending on the number of MBeans in the current WebLogic domain, this process make take a few minutes, and WLST may not return a prompt right away.

The `custom` command is available when WLST is connected to an Administration Server instance or a Managed Server instance. When connected to a WebLogic Integration or WebLogic Portal server, WLST can interact with all the WebLogic Integration or WebLogic Portal server MBeans.

For more information about custom MBeans, see *Developing Custom Management Utilities With JMX for Oracle WebLogic Server*.

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

Note: You can also navigate to custom MBeans on the Domain Runtime MBean Server using the `domainCustom()` command. See Section 3.11.3, "domainCustom," for more information.

3.11.1.2 Syntax

```
custom()
```

3.11.1.3 Example

The following example navigates from the configuration MBean hierarchy to the custom MBean hierarchy on a Administration Server instance.

```
wls:/mydomain/serverConfig> custom()
Location changed to custom tree. This is a writeable tree with No root. For more
help, use help('custom')
wls:/mydomain/custom>
```

3.11.2 domainConfig

Command Category: Tree Commands

Use with WLST: Online

3.11.2.1 Description

Navigates to the last MBean to which you navigated in the domain Configuration hierarchy or to the root of the hierarchy, `DomainMBean`. This read-only hierarchy stores the configuration MBeans that represent your current WebLogic domain.

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

3.11.2.2 Syntax

```
domainConfig()
```

3.11.2.3 Example

The following example navigates from the configuration MBean hierarchy to the WebLogic domain Configuration hierarchy on an Administration Server instance.

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

```
Location changed to domainConfig tree. This is a read-only tree with DomainMBean
as the root.
For more help, use help('domainConfig')
wls:/mydomain/domainConfig> ls()
dr--   AppDeployments
dr--   BridgeDestinations
dr--   Clusters
dr--   CustomResources
dr--   DeploymentConfiguration
dr--   Deployments
dr--   EmbeddedLDAP
dr--   ErrorHandlings
dr--   FileStores
dr--   InternalAppDeployments
dr--   InternalLibraries
dr--   JDBCDataSourceFactories
dr--   JDBCStores
dr--   JDBCSystemResources
dr--   JMSBridgeDestinations
dr--   JMSInteropModules
dr--   JMSServers
dr--   JMSSystemResources
...
wls:/mydomain/domainConfig>
```

3.11.3 domainCustom

Command Category: Tree Commands

Use with WLST: Online

3.11.3.1 Description

Navigates to the domain custom tree of custom MBeans that are registered in the Domain Runtime MBean Server. WLST navigates, interrogates, and edits domain custom MBeans as it does domain MBeans; however, domain custom MBeans cannot use the `cmo` variable because a stub is not available.

> **Note:** When navigating to the domainCustom tree, WLST queries all MBeans in the Domain Runtime MBean Server, the Runtime MBean Servers on each server, and potentially the JVM platform MBean server to locate the custom MBeans. Depending on the number of MBeans in the current WebLogic domain, this process make take a few minutes, and WLST may not return a prompt right away. It is recommended that a JMX query Object Name Pattern be specified to limit the amount of searching performed.

The `domainCustom` command is available only when WLST is connected to an Administration Server instance.

For more information about the Domain Runtime MBean Server, see "Understanding WebLogic Server MBeans" in *Developing Custom Management Utilities With JMX for Oracle WebLogic Server*.

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

3.11.3.2 Syntax

```
domainCustom(ObjectNamePattern)
```

Argument	Definition
ObjectNamePattern	A JMX query pattern, such as `sip:*`. The default value is null or `*:*`.

3.11.3.3 Example

The following example navigates from the configuration MBean hierarchy to the domain custom MBean hierarchy on an Administration Server instance:

```
wls:/mydomain/serverConfig> domainCustom()
Location changed to domain custom tree. This is a writeable tree with No root. For
more help, use help('domainCustom').

wls:/mydomain/domainCustom
```

3.11.4 domainRuntime

Command Category: Tree Commands

Use with WLST: Online

3.11.4.1 Description

Navigates to the last MBean to which you navigated in the domain Runtime hierarchy or to the root of the hierarchy, `DomainRuntimeMBean`. This read-only hierarchy stores the runtime MBeans that represent your current WebLogic domain.

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

3.11.4.2 Syntax

```
domainRuntime()
```

3.11.4.3 Example

The following example navigates from the configuration MBean hierarchy to the domain Runtime hierarchy on an Administration Server instance.

```
wls:/mydomain/serverConfig> domainRuntime()
wls:/mydomain/domainRuntime> ls()
dr--   AppRuntimeStateRuntime
dr--   DeployerRuntime
dr--   DomainServices
dr--   LogRuntime
dr--   MessageDrivenControlEJBRuntime
dr--   MigratableServiceCoordinatorRuntime
dr--   MigrationDataRuntimes
dr--   SNMPAgentRuntime
dr--   ServerLifeCycleRuntimes
dr--   ServerRuntimes
dr--   ServerServices

-r--   ActivationTime                          Mon Aug 01 11:41:25 EDT 2005
-r--   Clusters                                null
-r--   MigrationDataRuntimes                   null
-r--   Name                                    sampleMedRecDomain
-rw-   Parent                                  null
-r--   SNMPAgentRuntime                        null
```

```
-r--   Type                                          DomainRuntime
-r-x   restartSystemResource                         Void :
       WebLogicMBean(weblogic.management.configuration.SystemResourceMBean)
wls:/mydomain/domainRuntime>
```

3.11.5 edit

Command Category: Tree Commands

Use with WLST: Online

3.11.5.1 Description

Navigates to the last MBean to which you navigated in the edit configuration MBean hierarchy or to the root of the hierarchy, `DomainMBean`. This writable hierarchy stores all of the configuration MBeans that represent your current WebLogic domain.

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

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

3.11.5.2 Syntax

```
edit()
```

3.11.5.3 Example

The following example illustrates how to navigate from the server configuration MBean hierarchy to the editable copy of the domain configuration MBean hierarchy, in an Administration Server instance.

```
wls:/myserver/serverConfig> edit()
Location changed to edit tree. This is a writeable tree with DomainMBean as the
root.
For more help, use help('edit')
wls:/myserver/edit !> ls()
dr--   AppDeployments
dr--   BridgeDestinations
dr--   Clusters
dr--   DeploymentConfiguration
dr--   Deployments
dr--   EmbeddedLDAP
...
wls:/myserver/edit !>
```

3.11.6 jndi

Command Category: Tree Commands

Use with WLST: Online

3.11.6.1 Description

Navigates to the JNDI tree for the server to which WLST is currently connected. This read-only tree holds all the elements that are currently bound in JNDI.

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

3.11.6.2 Syntax

```
jndi()
```

3.11.6.3 Example

The following example navigates from the runtime MBean hierarchy to the Domain JNDI tree on an Administration Server instance.

```
wls:/myserver/runtime> jndi()
Location changed to jndi tree. This is a read-only tree with No root. For more
help, use help('jndi')
wls:/myserver/jndi> ls()
dr--   ejb
dr--   javax
dr--   jms
dr--   weblogic
...
```

3.11.7 serverConfig

Command Category: Tree Commands

Use with WLST: Online

3.11.7.1 Description

Navigates to the last MBean to which you navigated in the configuration MBean hierarchy or to the root of the hierarchy, `DomainMBean`.

This read-only hierarchy stores the configuration MBeans that represent the server to which WLST is currently connected. The MBean attribute values include any command-line overrides that a user specified while starting the server.

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

For more information, see "Navigating Among MBean Hierarchies" in *Oracle WebLogic Scripting Tool*.

3.11.7.2 Syntax

```
serverConfig()
```

3.11.7.3 Example

The following example navigates from the domain runtime MBean hierarchy to the configuration MBean hierarchy on an Administration Server instance.

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

3.11.8 serverRuntime

Command Category: Tree Commands

Use with WLST: Online

3.11.8.1 Description

Navigates to the last MBean to which you navigated in the runtime MBean hierarchy or to the root of the hierarchy, `ServerRuntimeMBean`. This read-only hierarchy stores the runtime MBeans that represent the server to which WLST is currently connected.

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

3.11.8.2 Syntax

```
serverRuntime()
```

3.11.8.3 Example

The following example navigates from the configuration MBean hierarchy to the runtime MBean hierarchy on an Administration Server instance.

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

3.12 WLST Variable Reference

Table 3–13 describes WLST variables and their common usage. All variables are initialized to default values at the start of a user session and are changed according to the user interaction with WLST.

Table 3–13 WLST Variables

Variable	Description	Example
cmgr	The cmgr variable is set to the ConfigurationManagerMBean. You can use this variable to get the current value of any ConfigurationManagerMBean attribute.	wls:/mydomain/edit> **cmgr.getCurrentEditor()** 'weblogic'
cmo	Current Management Object. The cmo variable is set to the bean instance to which you navigate using WLST. You can use this variable to perform any get, set, or invoke method on the current bean instance. 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 WebLogic domain to which WLST is connected.	wls:/mydomain/serverConfig> **print domainName** mydomain
domainRuntimeService	DomainRuntimeServiceMBean MBean. This variable is available only when WLST is connected to the Administration Server.	wls:/mydomain/serverConfig> **domainService.getServerName()** 'myserver'
editService	EditServiceMBean MBean. This variable is available only when WLST is connected to the Administration Server.	wls:/mydomain/edit> **dc = editService.getDomainConfiguration()**
exitonerror	Boolean value specifying whether WLST terminates script execution when it encounters an exception. This variable defaults to true, indicating that script execution is terminated when WLST encounters an error. This variable is not applicable when running WLST in interactive mode.	wls:/mydomain/serverConfig> **print exitonerror** true
home	Represents the local MBeanHome.	wls:/mydomain/serverConfig> **print home** weblogic.rmi.internal.BasicRemoteRef - hostID: '-*hostID*:[7001,7001,-1,-1,-1,-1,-1]:mydomain:AdminServer', oid: '260', channel: 'null'
isAdminServer	Boolean value specifying whether WLST is connected to a WebLogic Administration Server instance. WLST sets this variable to true if WLST is connected to a WebLogic Administration Server; otherwise, WLST sets it to false.	wls:/mydomain/serverConfig> **print isAdminServer** true

Table 3–13 (Cont.) WLST Variables

Variable	Description	Example
mbs	MBeanServerConnection object that corresponds to the current location in the hierarchy.	wls:/mydomain/serverConfig> **mbs.isRegistered(ObjectName('mydomain:Name=mydomain,Type=Domain'))**
recording	Boolean value specifying whether WLST is recording commands. WLST sets this variable to true when the startRecording command is entered; otherwise, WLST sets this variable to false.	wls:/mydomain/serverConfig> **print recording** true
runtimeService	RuntimeServiceMBean MBean.	wls:/mydomain/serverConfig> **sr=runtimeService.getServerRuntime()**
serverName	Name of the server to which WLST is connected.	wls:/mydomain/serverConfig> **print serverName** myserver
typeService	TypeServiceMBean MBean.	wls:/mydomain/serverConfig> **mi=typeService.getMBeanInfo('weblogic.management.configuration.ServerMBean')**
username	Name of user currently connected to WLST.	wls:/mydomain/serverConfig> **print username** weblogic
version	Current version of the running server to which WLST is connected.	wls:/mydomain/serverConfig> **print version** WebLogic Server 9.0 Thu Aug 31 12:15:50 PST 2005 778899

4

Infrastructure Security Custom WLST Commands

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

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

> **Note:** To use the Infrastructure Security custom WLST commands, you must invoke the WLST script from the Oracle Common home. 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

Table 4–1 (Cont.) WLST Command Categories

Command Category	Description
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.
Oracle Access Manager Commands	Manage OAM-related components, such as authorization providers, identity asserters, and SSO providers.

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.

Argument	Definition
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, componentType])
```

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for non-Java EE components.
componentType	Requests the audit policy for a specific component registered in the audit store. If not specified, the audit policy in `jps-config.xml` is returned.

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], [componentType],
[maxDirSize], [maxFileSize], [andCriteria], [orCriteria], [componentEventsFile])
```

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.
componentType	Specifies the component definition type to be updated. If not specified, the audit configuration defined in jps-config.xml is modified.
maxDirSize	Specifies the maximum size of the log directory.
maxFileSize	Specifies the maximum size of the log file.
andCriteria	Specifies the `and` criteria in a custom filter preset definition.
orCriteria	Specifies the `or` criteria in a custom filter preset definition.
componentEventsFile	Specifies a component definition file under the 11*g* Release 1 (11.1.1.6) metadata model. This parameter is required if you wish to create/update an audit policy in the audit store for an 11*g* Release 1 (11.1.1.6) metadata model component, and the filter preset level is set to "Custom".

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',removeCustomEvents='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 to limit the list to all events of 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, [componentType])
```

Argument	Definition
mbeanName	Specifies the name of the non-Java EE component MBean.
fileName	Specifies the path and file name to which the audit configuration should be exported.
componentType	Specifies that only events of the given component be exported to the file. If not specified, the audit configuration in `jps-config.xml` is 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=CSAuditProxyMBean',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, [componentType])
```

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.
componentType	Specifies that only events of the given component be imported from the file. If not specified, the audit configuration in `jps-config.xml` is 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='CSAuditProxyMBean',fileName='/tmp/auditconfig')
```

The following interactive command imports the audit configuration from a file; 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 key pair in a JKS keystore.	Online
getKeyStoreObject	Display a certificate or other object present in a JKS keystore.	Online

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

Use this command...	To...	Use with WLST...
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.

Argument	Definition
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.
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.

Argument	Definition
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.
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'.

Argument	Definition
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.
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.
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 server instance `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'.

Argument	Definition
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.

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 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

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

Use this command...	To...	Use with WLST...
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
deleteUserFederation	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
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

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

Use this command...	To...	Use with WLST...
getFederationPropertyList	Retrieve the federation property list.	Online
extractproviderprops	Export all provider configuration properties to a text file.	Script
setproviderprops	Set a provider's properties based on an input text file.	Script
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
oifStatus	Display the current status of Oracle Identity Federation on the managed server.	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
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.

Argument	Definition
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.

Argument	Definition
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 deleteUserFederation

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

```
deleteUserFederation([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> deleteUserFederation(['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.

Argument	Definition
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.
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 extractproviderprops

A WLST script that exports the properties of a provider.

4.4.36.1 Description

A WLST script that extracts all the configuration properties of the specified provider and exports them to a text file. You can later use this file to set the same properties on another provider. Execute this command from a UNIX or Windows command shell prompt and not from the WLST command shell. This script is stored in *ORACLE_HOME*`/fed/scripts`.

4.4.36.2 Syntax

```
extractproviderprops.py providerID filename
```

Argument	Definition
providerID	Specifies the name of the provider whose properties are to be extracted.
filename	Specifies the name of the text file to which the provider properties are extracted.

When you execute the script, you are prompted for the WebLogic administrator credentials and the connection URL; for the latter, specify the Managed Server port, not the Administration Server port.

File Format

The format of the extract file is:

```
TYPE:NAME:PROPNAME:PROPVALUE:PROPTYPE
```

For example:

```
X:X:sendattribute:false:boolean
MAP:attributelist/mailemail:datastore-attr:mail:string
LIST:sendattributefornameid:unspecified::string
```

4.4.37 setproviderprops

A WLST script that sets the properties of a provider using values from a text file.

4.4.37.1 Description

A WLST script that sets the properties of a provider using values from a text file. Execute this command from a UNIX or Windows command shell prompt and not from the WLST command shell. This script is stored in *ORACLE_HOME*/fed/scripts.

The text file is generated by the extractproviderprops command.

4.4.37.2 Syntax

```
setproviderprops.py providerID filename
```

Argument	Definition
providerID	Specifies the name of the provider whose properties are to be updated.
filename	Specifies the name of the input file from which to read the properties.

When you execute the script, you are prompted for the WebLogic administrator credentials and the connection URL; for the latter, specify the Managed Server port, not the Administration Server port.

4.4.38 getFederationPropertyMapEntry

Online command that returns a property value from a map.

4.4.38.1 Description

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

4.4.38.2 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.

4.4.38.3 Example

The following command returns property propA from a map:

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

4.4.39 listCustomAuthnEngines

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

4.4.39.1 Description

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

4.4.39.2 Syntax

```
listCustomAuthnEngines()
```

4.4.39.3 Example

The following command returns the list of all SP engines:

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

4.4.40 listCustomSPEngines

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

4.4.40.1 Description

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

4.4.40.2 Syntax

```
listCustomSPEngines()
```

4.4.40.3 Example

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

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

4.4.41 loadMetadata

Online command that loads metadata from an input file.

4.4.41.1 Description

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

4.4.41.2 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.

4.4.41.3 Example

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

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

4.4.42 oifStatus

Online command that reports the current status of the Oracle Identity Federation application in the managed server to which WLST is connected.

4.4.42.1 Description

This command displays the current status of Oracle Identity Federation on the managed server.

4.4.42.2 Syntax

```
loifStatus('serverurl', 'configfile', 'keyfile')
```

Argument	Definition
serverurl	Specifies the URL of the managed server.
configfile	This is a pre-defined user configuration file created with the WLST storeUserConfig command.
keyfile	This is a pre-defined key file created with the WLST storeUserConfig command

4.4.42.3 Example

The following command provides no arguments; WLST prompts you for the Oracle WebLogic Server username, password, and the managed server URL, then displays the federation server status:

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

The following command provides only the managed server URL; WLST prompts you for the Oracle WebLogic Server username and password:

```
wls:/mydomain/serverConfig> oifStatus('', '', 't3://localhost:7499')
```

The following command provides all arguments needed for WLST to display the federation server status:

```
wls:/mydomain/serverConfig> oifStatus('configfileA', 'keyfileB',
```

```
't3://localhost:7499')
```

4.4.43 removeConfigListInMap

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

4.4.43.1 Description

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

4.4.43.2 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.

4.4.43.3 Example

The following command removes the list property mylistA:

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

4.4.44 removeConfigMapEntryInMap

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

4.4.44.1 Description

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

4.4.44.2 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.

4.4.44.3 Example

The following command removes the nested property `myvarA`:

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

4.4.45 removeConfigMapInMap

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

4.4.45.1 Description

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

4.4.45.2 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.

4.4.45.3 Example

The following command removes the nested property myvarA:

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

4.4.46 removeConfigProperty

Online command that removes a configuration property.

4.4.46.1 Description

This command removes a property from config.xml.

4.4.46.2 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.

4.4.46.3 Example

The following command removes the property myvarA:

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

4.4.47 removeConfigPropertyList

Online command that removes a configuration property list.

4.4.47.1 Description

This command removes a property list from config.xml.

4.4.47.2 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.

4.4.47.3 Example

The following command removes the property list `mylistA`:

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

4.4.48 removeConfigPropertyMap

Online command that removes a property map.

4.4.48.1 Description

This command removes a property map in config.xml.

4.4.48.2 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.

4.4.48.3 Example

The following command removes mapA:

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

4.4.49 removeConfigPropertyMapEntry

Online command that removes a property value from a map.

4.4.49.1 Description

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

4.4.49.2 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.

4.4.49.3 Example

The following command removes property propA:

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

4.4.50 removeFederationListInMap

Online command that removes a property list in a map.

4.4.50.1 Description

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

4.4.50.2 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.

4.4.50.3 Example

The following command removes mylistA in mymapA:

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

4.4.51 removeFederationMapInMap

Online command that removes a nested map in a map.

4.4.51.1 Description

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

4.4.51.2 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.

4.4.51.3 Example

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

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

4.4.52 removeFederationMapEntryInMap

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

4.4.52.1 Description

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

4.4.52.2 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.

4.4.52.3 Example

The following command removes map property entry myvarA:

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

4.4.53 removeFederationProperty

Online command that removes a property value.

4.4.53.1 Description

This command removes a property entry in cot.xml.

4.4.53.2 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.

4.4.53.3 Example

The following command removes the provider property myvarA:

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

4.4.54 removeFederationPropertyList

Online command that removes a property list entry.

4.4.54.1 Description

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

4.4.54.2 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.

4.4.54.3 Example

The following command removes `mylistA`:

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

4.4.55 removeFederationPropertyMap

Online command that removes a property map.

4.4.55.1 Description

This command removes a property map in cot.xml.

4.4.55.2 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.

4.4.55.3 Example

The following command removes a map:

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

4.4.56 removeFederationPropertyMapEntry

Online command that removes a property value from a map.

4.4.56.1 Description

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

4.4.56.2 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.

4.4.56.3 Example

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

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

4.4.57 removePeerProviderEntry

Online command that removes a peer provider entry.

4.4.57.1 Description

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

4.4.57.2 Syntax

```
removePeerProviderEntry(providerID)
```

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

4.4.57.3 Example

The following command removes providerA:

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

4.4.58 setConfigProperty

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

4.4.58.1 Description

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

4.4.58.2 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.

4.4.58.3 Example

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

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

4.4.59 setCustomAuthnEngine

Online command that updates a custom authentication integration engine.

4.4.59.1 Description

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

4.4.59.2 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.

4.4.59.3 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')
```

4.4.60 setCustomSPEngine

Online command that updates a custom SP integration engine.

4.4.60.1 Description

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

4.4.60.2 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.

4.4.60.3 Example

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

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

4.4.61 setFederationProperty

Online command that adds or updates a property value.

4.4.61.1 Description

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

4.4.61.2 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.

4.4.61.3 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, to migrate policies and credentials from a source repository to a target repository, and to import and export (credential) encryption keys.

Table 4–5 WLST Security Commands

Use this command...	To...	Use with WLST...
listAppStripes	List application stripes in policy store.	Online
createAppRole	Create a new application role.	Online
deleteAppRole	Remove an application role.	Online
grantAppRole	Add a principal to a role.	Online

Table 4–5 (Cont.) WLST Security Commands

Use this command...	To...	Use with WLST...
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
addBootStrapCredential	Add a credential to the bootstrap credential store	Offline
exportEncryptionKey	Export the domain encryption key to the file `ewallet.p12`.	Offline
importEncryptionKey	Import the encryption key in file `ewallet.p12` to the domain.	Offline
restoreEncryptionKey	Restore the domain encryption key as it was before the last importing.	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
createResourceType	Create a new resource type.	Online
getResourceType	Fetch an existing resource type.	Online
deleteResourceType	Remove an existing resource type.	Online
createResource	Create a resource.	Online
deleteResource	Remove a resource.	Online
listResources	List resources in an application stripe.	Online
listResourceActions	List actions in a resource.	Online
createEntitlement	Create an entitlement.	Online
getEntitlement	List an entitlement.	Online
deleteEntitlement	Remove an entitlement.	Online
addResourceToEntitlement	Add a resource to an entitlement.	Online
revokeResourceFromEntitlement	Remove a resource from an entitlement	Online
listEntitlements	List entitlements in an application stripe.	Online

Table 4–5 (Cont.) WLST Security Commands

Use this command...	To...	Use with WLST...
grantEntitlement	Create an entitlement.	Online
revokeEntitlement	Remove an entitlement.	Online
listEntitlement	List an entitlement.	Online
listResourceTypes	List resource types in an application stripe.	Online

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,] [permActions])
```

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.
permActions	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", permActions="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,] [permActions])
```

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.

Argument	Definition
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.
permActions	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",
permActions="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,][preserveAppRoleGuid])
```

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][,preserveAppRoleGuid] [,mode])
```

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.

Argument	Definition
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.
preserveAppRoleGuid	Specifies whether the migration of policies should preserve or recreate GUIDs. Optional and false, by default. Set to true to preserve GUIDs; set to false to recreated GUIDs.
mode	Specifies whether the migration should stop and signal an error upon encountering a duplicate principal or a duplicate permission in an application policy. Set to lax to allow the migration to continue upon encountering duplicate items, to migrate just one of the duplicated items, and to log a warning to this effect; set to strict to force the migration to stop upon encountering duplicate items. If unspecified, it defaults to strict.

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 Example

The following invocation illustrates the migration of the file-based policies of application `PolicyServlet1` to file-based policies of application `PolicyServlet2`, that does not stop on encountering duplicate principals or permissions, that migrates just one of duplicate items, and that logs a warning when duplicates are found:

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

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>
```

For detailed configuration examples to use with this command, see *Oracle Fusion Middleware Security Guide*.

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).

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

4.6.13.3 Example

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 Example

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 Example

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 Example

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 addBootStrapCredential

Offline command that adds a credential to the bootstrap credential store.

4.6.17.1 Description

Adds a password credential with the given map, key, user name, and user password to the bootstrap credentials configured in the default JPS context of a JPS configuration file. In the event of an error, the command returns a `WLSTException`.

4.6.17.2 Syntax

```
addBootStrapCredential(jpsConfigFile, map, key, username, password)
```

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run.
map	Specifies the map of the credential to add.
key	Specifies the key of the credential to add.
username	Specifies the name of the user in the credential to add.
password	Specifies the password of the user in the credential to add.

4.6.17.3 Example

The following invocation adds a credential to the bootstrap credential store:

```
wls:/mydomain/serverConfig>
addBootStrapCredential(jpsConfigFile='./jps-config.xml', map='myMapName',
key='myKeyName', username='myUser', password='myPassword')
```

4.6.18 exportEncryptionKey

Offline command that extracts the encryption key from a domain's bootstrap wallet to the file `ewallet.p12`.

4.6.18.1 Description

Writes the domain's credential encryption key to the file `ewallet.p12`. The password passed must be used to import data from that file with the command `importEncryptionKey`.

4.6.18.2 Syntax

```
exportEncryptionKey(jpsConfigFile, keyFilePath, keyFilePassword)
```

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run.
keyFilePath	Specifies the directory where the file `ewallet.p12` is created; note that the content of this file is encrypted and secured by the value passed to `keyFilePassword`.
keyFilePassword	Specifies the password to secure the file `ewallet.p12`; note that this same password must be used when importing that file.

4.6.18.3 Example

The following invocation writes the file `ewallet.p12` in the directory myDir:

```
exportEncryptionKey(jpsConfigFile="pathName", keyFilePath="myDir"
,keyFilePassword="password")
```

4.6.19 importEncryptionKey

Offline command that imports keys from the specified ewallet.p12 file into the domain.

4.6.19.1 Description

Imports encryption keys from the file `ewallet.p12` into the domain. The password passed must be the same as that used to create the file with the command `exportEncryptionKey`.

4.6.19.2 Syntax

```
importEncryptionKey(jpsConfigFile, keyFilePath, keyFilePassword)
```

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run.
keyFilePath	Specifies the directory where the `ewallet.p12` is located.
keyFilePassword	Specifies the password used when the file `ewallet.p12` was generated.

4.6.19.3 Example

```
importEncryptionKey(jpsConfigFile="pathName", keyFilePath="dirloc"
```

```
,keyFilePassword="password")
```

4.6.20 restoreEncryptionKey

Offline command to restore the domain credential encryption key.

4.6.20.1 Description

Restores the state of the domain bootstrap keys as it was before running importEncryptionKey.

4.6.20.2 Syntax

```
restoreEncryptionKey(jpsConfigFile)
```

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run.

4.6.20.3 Example

```
restoreEncryptionKey(jpsConfigFile="pathName")
```

4.6.21 reassociateSecurityStore

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

4.6.21.1 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. This command also allows setting up a policy store shared by different domains (see optional argument `join` below). In the event of an error, the command returns a `WLSTException`. This command runs in interactive mode only.

4.6.21.2 Syntax

```
reassociateSecurityStore(domain, admin, password, ldapurl, servertype, jpsroot [,
join] [,keyFilePath, keyFilePassword])
```

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.

Argument	Definition
jpsroot	Specifies the root node in the target LDAP repository under which all data is migrated. The format is `cn=nodeName`.
join	Specifies whether the domain is to share a policy store specified in some other domain. Optional. Set to true to share an existing policy store in another domain; set to false otherwise. If unspecified, it defaults to false. The use of this argument allows multiple WebLogic domains to point to the same logical policy store.
keyFilePath	Specifies the directory where the `ewallet.p12` is located.
keyFilePassword	Specifies the password used when the file `ewallet.p12` was generated.

4.6.21.3 Examples

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")
```

Suppose that you want some *other* domain (distinct from `myDomain`, say `otherDomain`) to share the policy store in `myDomain`. Then you would invoke the command as follows:

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

4.6.22 upgradeSecurityStore

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

4.6.22.1 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`.

4.6.22.2 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.

4.6.22.3 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)
```

4.6.23 createResourceType

Online command that creates a new resource type in the domain policy store within a given application stripe.

4.6.23.1 Description

Creates a new resource type element in the domain policy store within a given application stripe and with specified name, display name, description, and actions.

Optional arguments are enclosed in between square brackets; all other arguments are required. In the event of an error, the command returns a `WLSTException`.

4.6.23.2 Syntax

Optional arguments are enclosed in square brackets.

```
createResourceType(appStripe, resourceTypeName, displayName, description [,
provider] [, matcher], actions [, delimeter])
```

Argument	Definition
appStripe	Specifies the application stripe where to insert the resource type.
resourceTypeName	Specifies the name of the resource type to insert.
displayName	Specifies the name for the resource type used in UI gadgets.
description	Specifies a brief description of the resource type.
provider	Specifies the provider for the resource type.
matchere	Specifies the class of the resource type. If unspecified, it defaults to `oracle.security.jps.ResourcePermission`.
actions	Specifies the actions allowed on instances of the resource type.
delimeter	Specifies the character used to delimit the list of actions. If unspecified, it defaults to comma ','.

4.6.23.3 Example

The following invocation creates a resource type in the stripe myApplication with actions BWPrint and ColorPrint delimited by a semicolon:

```
wls:/mydomain/serverConfig> createResourceType(appStripe="myApplication",
resourceTypeName="resTypeName", displayName="displName", description="A resource
type", provider="Printer", matcher="com.printer.Printer",
actions="BWPrint;ColorPrint" [, delimeter=";"])
```

4.6.24 getResourceType

Online command that fetches a resource type from the domain policy store within a given application stripe.

4.6.24.1 Description

Gets the relevant parameters of a <resource-type> entry in the domain policy store within a given application stripe and with specified name. In the event of an error, the command returns a `WLSTException`.

4.6.24.2 Syntax

```
getResourceType(appStripe, resourceTypeName)
```

Argument	Definition
appStripe	Specifies the application stripe from where to fetch the resource type.
resourceTypeName	Specifies the name of the resource type to fetch.

4.6.24.3 Example

The following invocation fetches the resource type myResType from the stripe myApplication:

```
wls:/mydomain/serverConfig> getResourceType(appStripe="myApplication",
resourceTypeName="myResType")
```

4.6.25 deleteResourceType

Online command that removes a resource type from the domain policy store within a given application stripe.

4.6.25.1 Description

Removes a <resource-type> entry in the domain policy store within a given application stripe and with specified name. In the event of an error, the command returns a `WLSTException`.

4.6.25.2 Syntax

```
deleteResourceType(appStripe, resourceTypeName)
```

Argument	Definition
appStripe	Specifies the application stripe from where to remove the resource type.
resourceTypeName	Specifies the name of the resource type to remove.

4.6.25.3 Example

The following invocation removes the resource type myResType from the stripe myApplication:

```
wls:/mydomain/serverConfig> deleteResourceType(appStripe="myApplication",
resourceTypeName="myResType")
```

4.6.26 listAppStripes

Online or offline command that lists the application stripes in the policy store.

4.6.26.1 Description

This script can be run in offline or online mode. When run in offline mode, a configuration file must be passed, and it lists the application stripes in the policy store referred to by the configuration in the default context of the passed configuration file; the default configuration *must not* have a service instance reference to an identity store. When run in online mode, a configuration file must not be passed, and it lists stripes in the policy store of the domain to which you connect. In any mode, if a regular expression is passed, it lists the application stripes with names that match the regular expression; otherwise, it lists all application stripes.

If this command is used in offline mode after reassociating to a DB-based store, the configuration file produced by the reassociation *must* be manually edited as described in "Running listAppStripes after Reassociating to a DB-Based Store" in *Oracle Fusion Middleware Security Guide*.

4.6.26.2 Syntax

```
listAppStripes([configFile="configFileName"] [, regularExpression="aRegExp"])
```

Argument	Definition
configFile	Specifies the path to the OPSS configuration file. Optional. If specified, the script runs offline; the default context in the specified configuration file *must not* have a service instance reference to an identity store. If unspecified, the script runs online and it lists application stripes in the policy store.
regularExpression	Specifies the regular expression that returned stripe names should match. Optional. If unspecified, it matches all names. To match substrings, use the character *.

4.6.26.3 Examples

The following (online) invocation returns the list of application stripes in the policy store:

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

The following (offline) invocation returns the list of application stripes in the policy store referenced in the default context of the specified configuration file:

```
wls:/mydomain/serverConfig> listAppStripes(configFile="
/home/myFile/jps-config.xml")
```

The following (online) invocation returns the list of application stripes that contain the prefix App:

```
wls:/mydomain/serverConfig> listAppStripes(regularExpression="App*")
```

4.6.27 createResource

Online command that creates a new resource.

4.6.27.1 Description

Creates a resource of a specified type in a specified application stripe. The passed resource type must exist in the passed application stripe.

4.6.27.2 Syntax

```
createResource(appStripe="appStripeName", name="resName", type="resTypeName"
[,-displayName="dispName"] [,-description="descript"])
```

Argument	Definition
appStripe	Specifies the application stripe where the resource is created.
name	Specifies the name of the resource created.
type	Specifies the type of resource created. The passed resource type *must* be present in the application stripe at the time this script is invoked.
displayName	Specifies the display name of the resource created. Optional.
description	Specifies the description of the resource created. Optional.

4.6.27.3 Example

The following invocation creates the resource myResource in the stripe myApplication:

```
wls:/mydomain/serverConfig> createResource(appStripe="myApplication",
name="myResource", type="myResType", displayName="myNewResource")
```

4.6.28 deleteResource

Online command that deletes a resource.

4.6.28.1 Description

Deletes a resource and all its references from entitlements in an application stripe. It performs a cascading deletion: if the entitlement refers to one resource only, it removes the entitlement; otherwise, it removes from the entitlement the resource actions for the passed type.

4.6.28.2 Syntax

```
deleteResource(appStripe="appStripeName", name="resName", type="resTypeName")
```

Argument	Definition
appStripe	Specifies the application stripe where the resource is deleted.
name	Specifies the name of the resource deleted.
type	Specifies the type of resource deleted. The passed resource type *must* be present in the application stripe at the time this script is invoked.

4.6.28.3 Example

The following invocation deletes the resource myResource in the stripe myApplication:

```
wls:/mydomain/serverConfig> deleteResource(appStripe="myApplication",
name="myResource", type="myResType")
```

4.6.29 listResources

Online command that lists resources in a specified application stripe.

4.6.29.1 Description

If a resource type is specified, it lists all the resources of the specified resource type; otherwise, it lists all the resources of all types.

4.6.29.2 Syntax

```
listResources(appStripe="appStripeName" [,type="resTypeName"])
```

Argument	Definition
appStripe	Specifies the application stripe where the resources are listed.
type	Specifies the type of resource listed. The passed resource type *must* be present in the application stripe at the time this script is invoked.

4.6.29.3 Example

The following invocation lists all resources of type myResType in the stripe myApplication:

```
wls:/mydomain/serverConfig> listResources(appStripe="myApplication",
type="myResType")
```

4.6.30 listResourceActions

Online command that lists the resources and actions in an entitlement.

4.6.30.1 Description

Lists the resources and actions in an entitlement within an application stripe.

4.6.30.2 Syntax

```
listResourceActions(appStripe="appStripeName", permSetName="entitlementName")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement resides.
permSetName	Specifies the name of the entitlement whose resources and actions to list.

4.6.30.3 Example

The following invocation lists the resources and actions of the entitlement myEntitlement in the stripe myApplication:

```
wls:/mydomain/serverConfig> listResourceActions(appStripe="myApplication",
permSetName="myEntitlement")
```

4.6.31 createEntitlement

Online command that creates a new entitlement.

4.6.31.1 Description

Creates a new entitlement with just one resource and a list of actions in a specified application stripe. Use `addResourceToEntitlement` to add additional resources to an existing entitlement; use `revokeResourceFromEntitlement` to delete resources from an existing entitlement.

4.6.31.2 Syntax

```
createEntitlement(appStripe="appStripeName", name="entitlementName",
resourceName="resName", actions="actionList" [,-displayName="dispName"]
[,-description="descript"])
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is created.
name	Specifies the name of the entitlement created.
resourceName	Specifies the name of the one resource member of the entitlement created.
actions	Specifies a comma-separated the list of actions for the resource resourceName.
displayName	Specifies the display name of the resource created. Optional.
description	Specifies the description of the entitlement created. Optional.

4.6.31.3 Example

The following invocation creates the entitlement myEntitlement with just the resource myResource in the stripe myApplication:

```
wls:/mydomain/serverConfig> createEntitlement(appStripe="myApplication",
name="myEntitlement", resourceName="myResource", actions="read,write")
```

4.6.32 getEntitlement

Online command that gets an entitlement.

4.6.32.1 Description

Returns the name, display name, and all the resources (with their actions) of an entitlement in an application stripe.

4.6.32.2 Syntax

```
getEntitlement(appStripe="appStripeName", name="entitlementName")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is located.
name	Specifies the name of the entitlement to access.

4.6.32.3 Example

The following invocation returns the information of the entitlement myEntitlement in the stripe myApplication:

```
wls:/mydomain/serverConfig> getEntitlement(appStripe="myApplication",
name="myEntitlement")
```

4.6.33 deleteEntitlement

Online command that deletes an entitlement.

4.6.33.1 Description

Deletes an entitlement in a specified application stripe. It performs a cascading deletion by removing all references to the specified entitlement in the application stripe.

4.6.33.2 Syntax

```
deleteEntitlement(appStripe="appStripeName", name="entitlementName")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is deleted.
name	Specifies the name of the entitlement to delete.

4.6.33.3 Example

The following invocation deletes the entitlement myEntitlement in the stripe myApplication:

```
wls:/mydomain/serverConfig> deleteEntitlement(appStripe="myApplication",
name="myEntitlement")
```

4.6.34 addResourceToEntitlement

Online command that adds a resource with specified actions to an entitlement.

4.6.34.1 Description

Adds a resource with specified actions to an entitlement in a specified application stripe. The passed resource type must exist in the passed application stripe.

4.6.34.2 Syntax

```
addResourceToEntitlement(appStripe="appStripeName", name="entName",
resourceName="resName",actions="actionList")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is located.
name	Specifies the name of the entitlement to modify.
resourceName	Specifies the name of the resource to add.
resourceType	Specifies the type of the resource to add. The passed resource type *must* be present in the application stripe at the time this script is invoked.
actions	Specifies the comma-separated list of actions for the added resource.

4.6.34.3 Example

The following invocation adds the resource myResource to the entitlement myEntitlement in the application stripe myApplication:

```
wls:/mydomain/serverConfig> addResourceToEntitlement(appStripe="myApplication",
name="myEntitlement", resourceName="myResource", resourceType="myResType",
actions="view,edit")
```

4.6.35 revokeResourceFromEntitlement

Online command that removes a resource from an entitlement.

4.6.35.1 Description

Removes a resource from an entitlement in a specified application stripe.

4.6.35.2 Syntax

```
revokeResourceFromEntitlement(appStripe="appStripeName", name="entName",
resourceName="resName", resourceType="resTypeName", actions="actionList")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is located.
name	Specifies the name of the entitlement to modify.
resourceName	Specifies the name of the resource to remove.
resourceType	Specifies the type of the resource to remove.
actions	Specifies the comma-separated list of actions to remove.

4.6.35.3 Example

The following invocation removes the resource myResource from the entitlement myEntitlement in the stripe myApplication:

```
wls:/mydomain/serverConfig>
revokeResourceFromEntitlement(appStripe="myApplication", name="myEntitlement",
```

```
resourceName="myResource", resourceType="myResType", actions="view,edit")
```

4.6.36 listEntitlements

Online command that lists the entitlements in an application stripe.

4.6.36.1 Description

Lists all the entitlements in an application stripe. If a resource name and a resource type are specified, it lists the entitlements that have a resource of the specified type matching the specified resource name; otherwise, it lists all the entitlements in the application stripe.

4.6.36.2 Syntax

```
listEntitlements(appStripe="appStripeName" [,resourceTypeName="resTypeName",
resourceName="resName"])
```

Argument	Definition
appStripe	Specifies the application stripe from where to list entitlements.
resourceTypeName	Specifies the name of the type of the resources to list. Optional.
resourceName	Specifies the name of resource to match. Optional.

4.6.36.3 Examples

The following invocation lists all the entitlements in the stripe myApplication:

```
wls:/mydomain/serverConfig> listEntitlements(appStripe="myApplication")
```

The following invocation lists all the entitlements in the stripe myApplication that contain a resource type myResType and a resource whose name match the resource name myResName:

```
wls:/mydomain/serverConfig> listEntitlements(appStripe="myApplication",
resourceTypeName="myResType", resourceName="myResName")
```

4.6.37 grantEntitlement

Online command that creates a new entitlement.

4.6.37.1 Description

Creates a new entitlement with a specified principal in a specified application stripe.

4.6.37.2 Syntax

```
grantEntitlement(appStripe="appStripeName", principalClass="principalClass",
principalName="principalName" ,-permSetName="entName")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is created.
principalClass	Specifies the class associated with the principal.
principalName	Specifies the name of the principal to which the entitlement is granted.
permSetName	Specifies the name of the entitlement created.

4.6.37.3 Example

The following invocation creates the entitlement myEntitlement in the stripe myApplication:

```
wls:/mydomain/serverConfig> grantEntitlement(appStripe="myApplication",
principalClass="oracle.security.jps.service.policystore.ApplicationRole",
principalName="myPrincipalName", permSetName="myEntitlement")
```

4.6.38 revokeEntitlement

Online command that deletes an entitlement.

4.6.38.1 Description

Deletes an entitlement and revokes the entitlement from the principal in a specified application stripe.

4.6.38.2 Syntax

```
revokeEntitlement(appStripe="appStripeName", principalClass="principalClass",
principalName="principalName" ,-permSetName="entName")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is deleted.
principalClass	Specifies the class associated with the principal.
principalName	Specifies the name of the principal to which the entitlement is revoked.
permSetName	Specifies the name of the entitlement deleted.

4.6.38.3 Example

The following invocation deleted the entitlement myEntitlement in the stripe myApplication:

```
wls:/mydomain/serverConfig> revokeEntitlement(appStripe="myApplication",
principalClass="oracle.security.jps.service.policystore.ApplicationRole",
principalName="myPrincipalName", permSetName="myEntitlement")
```

4.6.39 listEntitlement

Online command that lists an entitlement in a specified application stripe.

4.6.39.1 Description

If a principal name and a class are specified, it lists the entitlements that match the specified principal; otherwise, it lists all the entitlements.

4.6.39.2 Syntax

```
listEntitlement(appStripe="appStripeName" [, principalName="principalName",
principalClass="principalClass"])
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is deleted.
principalName	Specifies the name of the principal to match. Optional.
principalClass	Specifies the class of the principal to match. Optional.

4.6.39.3 Example

The following invocation lists all entitlements in the stripe myApplication:

```
wls:/mydomain/serverConfig> listEntitlement(appStripe="myApplication")
```

4.6.40 listResourceTypes

Online command that lists resource types.

4.6.40.1 Description

Lists all the resource types in a specified application stripe.

4.6.40.2 Syntax

```
listResourceTypes(appStripe="appStripeName")
```

Argument	Definition
appStripe	Specifies the application stripe where the resource types are located.

4.6.40.3 Example

The following invocation lists all resource types in the stripe myApplication:

```
wls:/mydomain/serverConfig> listEntitlement(appStripe="myApplication")
```

4.7 Oracle Access Manager Commands

Use the WLST commands listed in Table 4–6 to manage Oracle Access Manager (OAM)-related components, such as authorization providers, identity asserters, and SSO providers, as well as to display metrics and deployment topology, manage Oracle Access Manager server and agent configuration and logger settings.

Table 4–6 WLST Oracle Access Manager Commands

Use this command...	To...	Use with WLST...
listOAMAuthnProviderParams	List the parameters set for an Oracle Access Manager authentication or identity assertion provider.	Online
createOAMIdentityAsserter	Create a new identity asserter.	Online
updateOAMIdentityAsserter	Update an existing identity asserter.	Online
createOAMAuthenticator	Create a new authenticator.	Online
deleteOAMAuthnProvider	Delete an existing authentication provider.	Online
updateOAMAuthenticator	Update an existing authenticator.	Online
addOAMSSOProvider	Add a new SSO provider.	Online
displayTopology	List the details of deployed Oracle Access Manager Servers.	Online Offline
displayMetrics	Display the performance metrics of an Oracle Access Manager Server and domain	Online
displayOamServer	Display Oracle Access Manager Server configuration details.	Online Offline

Table 4–6 (Cont.) WLST Oracle Access Manager Commands

Use this command...	To...	Use with WLST...
createOamServer	Create an entry for an Oracle Access Manager Server configuration.	Online Offline
editOamServer	Edit the entry for an Oracle Access Manager Server configuration.	Online Offline
deleteOamServer	Delete the named Oracle Access Manager Server configuration.	Online Offline
displayOssoAgent	Display OSSO Agent configuration details.	Online Offline
editOssoAgent	Edit OSSO Agent configuration details.	Online Offline
deleteOssoAgent	Delete the named OSSO Agent configuration.	Online Offline
displayWebgateAgent	Display WebGate Agent configuration details.	Online Offline
editWebgateAgent	Edit 10g WebGate Agent registration details.	Online Offline
deleteWebgateAgent	Delete the named 10g WebGate Agent configuration.	Online Offline
changeLoggerSetting	Change Logger Settings.	Online Offline
changeConfigDataEncryptionKey	Regenerate the configuration data encryption key and re-encrypt data.	Online Offline
displayUserIdentityStore	Display a user identity store registration.	Online Offline
editUserIdentityStore	Edit a user identity store registration.	Online Offline
createUserIdentityStore	Create a user identity store registration.	Online Offline
deleteUserIdentityStore	Delete a user identity store registration.	Online Offline
configRequestCacheType	Configure the SSO server request cache type.	Online Offline
displayRequestCacheType	Display the SSO server request cache type entry.	Online Offline
exportPolicy	Export Oracle Access Manager policy data from a test (source) to an intermediate Oracle Access Manager file.	Online
importPolicy	Import Oracle Access Manager policy data from the Oracle Access Manager file specified.	Online

Table 4–6 (Cont.) WLST Oracle Access Manager Commands

Use this command...	To...	Use with WLST...
importPolicyDelta	Import Oracle Access Manager policy changes from the Oracle Access Manager file specified.	Online
migratePartnersToProd	Migrate partners from the source Oracle Access Manager Server to the specified target Oracle Access Manager Server.	Online
exportPartners	Export the Oracle Access Manager partners from the source to the intermediate Oracle Access Manager file specified.	Online
importPartners	Import the Oracle Access Manager partners from the intermediate Oracle Access Manager file specified.	Online
configureOAAM	Configure the Oracle Access Manager-Oracle Adaptive Access Manager basic integration.	Online
registerOIFDAPPartner	Register Oracle Identity Federation as Delegated Authentication Protocol (DAP) Partner.	Online Offline
enableCoexistMode	Enable the Coexist Mode.	Online
disableCoexistMode	Disable the Coexist Mode.	Online
editGITOValues	Edit GITO configuration parameters.	Online Offline
editWebgate11gAgent	Edit an 11g WebGate registration.	Online Offline
deleteWebgate11gAgent	Remove an 11g WebGate Agent registration.	Online Offline
displayWebgate11gAgent	Display an 11g WebGate Agent registration.	Online Offline
displayOAMMetrics	Display metrics of OAM Servers.	Online Offline
updateOIMHostPort	Update the Oracle Identity Manager configuration when integrated with Oracle Access Manager.	Online Offline
configureOIM	Creates an Agent registration specific to Oracle Identity Manager when integrated with Oracle Access Manager.	Online
updateOSSOResponseCookieConfig	Updates OSSO Proxy response cookie settings.	Online Offline
deleteOSSOResponseCookieConfig	Deletes OSSO Proxy response cookie settings.	Online Offline
displaySimpleModeGlobalPassphrase	Displays the simple mode global passphrase in plain text from the system configuration.	Online
exportSelectedPartners	Exports selected OAM Partners to the intermediate OAM file specified.	Online
migrateArtifacts	Migrates artifacts based on the input artifact file.	Online

Table 4–6 (Cont.) WLST Oracle Access Manager Commands

Use this command...	To...	Use with WLST...
registerThirdPartyTAPPartner	Registers any third party as a Trusted Authentication Protocol (TAP) Partner.	Online

4.7.1 listOAMAuthnProviderParams

Online command that lists the values of the parameters in effect in a domain authenticator or identity asserter.

4.7.1.1 Description

Lists the values of the parameters set for a given Oracle Access Manager authenticator or identity asserter. In the event of an error, the command returns a `WLSTException`.

4.7.1.2 Syntax

```
listOAMAuthnProviderParams(name)
```

Argument	Definition
name	Specifies the name of the authenticator or identity asserter.

4.7.1.3 Example

The following invocation lists the parameters and values set for the asserter named `myIdAsserter`:

```
listOAMAuthnProviderParams(name="myIdAsserter")
```

4.7.2 createOAMIdentityAsserter

Online command that creates an Oracle Access Manager identity asserter in the current domain.

4.7.2.1 Description

Creates an identity asserter with a given name in the current domain. Before executing this command, make sure that no Oracle Access Manager identity asserter is already configured in the current domain. In the event of an error, the command returns a `WLSTException`.

4.7.2.2 Syntax

```
createOAMIdentityAsserter(name)
```

Argument	Definition
name	Specifies the name of the new identity asserter. If no name is specified, it defaults to "OAMIdentityAsserter".

4.7.2.3 Example

The following invocation creates a new identity asserter named `OAMIdentityAsserter`:

```
createOAMIdentityAsserter(name="OAMIdentityAsserter")
```

4.7.3 updateOAMIdentityAsserter

Online command that updates the values of parameters of the Oracle Access Manager identity asserter in the current domain.

4.7.3.1 Description

Updates the value of given parameters of the domain Oracle Access Manager identity asserter. In the event of an error, the command returns a `WLSTException`.

4.7.3.2 Syntax

`updateOAMIdentityAsserter(name, paramNameValueList)`

Argument	Definition
name	Specifies the name of the Oracle Access Manager identity asserter whose parameter values to update.
paramNameValueList	Specifies the comma-separated list of pairs of parameter name-value to be updated. The format of each pair is: `paramName="paramValue"` The parameter names that can be updated are the following only: ■ `accessGateName`—The name of the AccessGate used by the authenticator. ■ `accessGatePwd`—The password to the AccessGate used by the authenticator. ■ `pAccessServer`—The name of the primary access server. Values must have the format hostName:portNumber. ■ `sAccessServer`—The name of the secondary access server. Values must have the format hostName:portNumber. ■ `transportSecurity`—The mode of communication between AccessGate and OAM Access Server. ■ `keystorePwd`—The password to access the domain key store. ■ `keystorePath`—The absolute path of the JKS key store used for SSL communication between the authenticator and OAM Access Server. ■ `simpleModePassphrase`—The password shared by AccessGate and OAM Access Server in simple communication mode. ■ `truststorePath` —The absolute path of the JKS trust store used for SSL communication between the authenticator and OAM Access Server. ■ `poolMaxConnections`—The maximum number of connections in the OAM Server connection pool. ■ `poolMinConnections`—The minimum number of connections in the OAM Server connection pool. ■ `ssoHeaderName`—The SSO header name. ■ `controlFlag`—The JAAS control flag that sets up dependencies among all authenticators in the domain. Values can be only REQUIRED, SUFFICIENT, REQUISITE, or OPTIONAL. ■ `appDomain`—The name of the application domain.

4.7.3.3 Example

The following invocation updates the parameters `accessGateName`, `accessGatePwd`, `pAccessServer`, and `ssoHeaderName` in the Oracle Access Manager identity asserter named `myIdAsserter`:

```
updateOAMIdentityAsserter(name="myIdAsserter",
accessGateName="OAM IAP AccessGate", accessGatePwd="welcome1",
pAccessServer="myhost.domain.com:5543", ssoHeaderName="OAM_SSO_HEADER")
```

4.7.4 createOAMAuthenticator

Online command that creates an Oracle Access Manager authenticator in the current domain.

4.7.4.1 Description

Creates an Oracle Access Manager authenticator with a given name in the current domain. Before executing this command, make sure that no Oracle Access Manager authenticator is already configured in the default security domain. In the event of an error, the command returns a `WLSTException`.

4.7.4.2 Syntax

```
createOAMAuthenticator(name)
```

Argument	Definition
name	Specifies the name of the new authentication provider in the default domain. If no name is specified, it defaults to "OAMAuthenticator".

4.7.4.3 Example

The following invocation creates a new authentication provider named `OAMAuthenticator`:

```
createOAMAuthenticator(name="OAMAuthenticator")
```

4.7.5 deleteOAMAuthnProvider

Online command that deletes the OAM authenticator from the current domain.

4.7.5.1 Description

Deletes the OAM authenticator with a given name from the current domain. In the event of an error, the command returns a `WLSTException`.

4.7.5.2 Syntax

```
deleteOAMAuthnProvider(name)
```

Argument	Definition
name	Specifies the name of the authentication provider to delete.

4.7.5.3 Example

The following invocation deletes the authenticator `myAuthenticator`:

```
deleteOAMAuthnProvider(name="myAuthenticator")
```

4.7.6 updateOAMAuthenticator

Online command that updates the values of parameters of the Oracle Access Manager authenticator in the current domain.

4.7.6.1 Description

Updates the value of given parameters of the domain Oracle Access Manager authenticator. In the event of an error, the command returns a `WLSTException`.

4.7.6.2 Syntax

`updateOAMAuthenticator(name, paramNameValueList)`

Argument	Definition
name	Specifies the name of the Oracle Access Manager authenticator whose parameter values to update.
paramNameValueList	Specifies the comma-separated list of pairs of parameter name-value to be updated. The format of each pair is `paramName='paramValue'` The only parameter names that can be updated are the following: ■ `accessGateName`—The name of the AccessGate used by the authenticator. ■ `accessGatePwd`—The password to the AccessGate used by the authenticator. ■ `pAccessServer`—The name of the primary access server. Values must have the format hostName:portNumber. ■ `sAccessServer`—The name of the secondary access server. Values must have the format hostName:portNumber. ■ `transportSecurity`—The mode of communication between AccessGate and OAM Access Server: open, simple, or cert. ■ `keystorePwd`—The password to access the domain key store. ■ `keystorePath`—The absolute path of the JKS key store used for SSL communication between the authenticator and OAM Access Server. ■ `simpleModePassphrase`—The password shared by AccessGate and OAM Access Server in simple communication mode. ■ `truststorePath`—The absolute path of the JKS trust store used for SSL communication between the authenticator and OAM Access Server. ■ `poolMaxConnections`—The maximum number of connections in the OAM Server connection pool. ■ `poolMinConnections`—The minimum number of connections in the OAM Server connection pool. ■ `useRetNameAsPrincipal`—Specifies whether the user name retrieved from the OAM authenticator should be used as the name of the Principal in the Subject. ■ `controlFlag`—The JAAS control flag that sets up dependencies among all authenticators in the domain. Values can be only REQUIRED, SUFFICIENT, REQUISITE, or OPTIONAL. ■ `appDomain`—The name of the application domain.

4.7.6.3 Example

The following invocation updates the parameters `accessGateName`, `accessGatePwd`, and `pAccessServer` in the Oracle Access Manager authenticator named `myAuthenticator`:

```
updateOAMAuthenticator(name="myAuthenticator",
accessGateName="OAM AP AccessGate", accessGatePwd="welcome1",
pAccessServer="myhost.domain.com:5543")
```

4.7.7 addOAMSSOProvider

Online command that adds an Oracle Access Manager SSO provider with the given login URI, logout URI, and auto-login URI.

4.7.7.1 Description

Adds an SSO provider with the given login URI, logout URI, and auto-login URI. This command modifies the domain `jps-config.xml` by adding an Oracle Access Manager SSO service instance with the required properties. In the event of an error, the command returns a `WLSTException`.

4.7.7.2 Syntax

```
addOAMSSOProvider(loginuri, logouturi, autologinuri, beginimpuri, endimpuri)
```

Argument	Definition
loginuri	Required. Specifies the URI of the login page and triggers SSO authentication.
logouturi	Optional. Specifies the URI of the logout page and logs the signed-on user out. If unspecified, defaults to `logouturi=NONE`. Set to "" to ensure that ADF security calls the OPSS logout service, which uses the implementation of the class `OAMSSOServiceImpl` to clear the cookie `ObSSOCookie`. More generally, an ADF-secured web application that would like to clear cookies without logging out the user should use this setting.
autologinuri	Required. Specifies the URI of the autologin page. Optional. If unspecified, it defaults to `autologin=NONE`.
beginimpuri	Optional. Specifies the URI that triggers the impersonation SSO session.
endimpuri	Optional. Specifies the URI that terminates the impersonation SSO session.

4.7.7.3 Example

The following invocation adds an SSO provider with the passed URIs; note the special behavior implied by the setting `logouturi=""` and the impersonation parameters, as explained in the above table:

```
addOAMSSOProvider(loginuri="/${app.context}/adfAuthentication",
logouturi="/oamsso/logout.html",
beginimpuri="https://login.acme.com/impersonationInit.html"
endimpuri="https://login.acme.com/impersonationTerm.html")
autologin="/fooBar.cgi")
```

4.7.8 displayTopology

Online and offline command that displays the information about all the OAM Servers in a deployment.

4.7.8.1 Description

Lists the topology of deployed OAM Servers. There are no arguments for this command.

4.7.8.2 Syntax

```
displayTopology
```

4.7.8.3 Example

The following invocation lists the details of all deployed OAM Servers, as described above:

```
displayTopology
```

4.7.9 displayMetrics

Online command that displays the performance metrics of an OAM Server and domain.

4.7.9.1 Description

Displays the performance metrics of an OAM Server and domain specific to collectors, including host, process, and server names. There are no arguments for this command.

If none of the arguments are specified all the details of all the servers and collectors are displayed.

4.7.9.2 Syntax

```
displayMetrics()
```

4.7.9.3 Example

The following invocation lists all metrics specific to named collectors, as described above:

```
displayMetrics()
```

4.7.10 displayOamServer

Online and offline command that displays OAM Server registration details.

4.7.10.1 Description

Displays OAM Server registration details, including the host, port, registration name, OAM Proxy port and server ID, and, optionally, the OAM Proxy shared secret.

The scope of this command is an instance, only. The scope is not an argument.

4.7.10.2 Syntax

```
displayOamServer(host,port)
```

Argument	Definition
host	Mandatory. Specifies the name of the OAM Server host.
port	Mandatory. Specifies the listening port of the OAM Server host.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.10.3 Example

The following invocation lists all metrics specific to named collectors, as described above:

```
displayOamServer(host="my_host", port="15000", domainHome="domainHome1")
```

4.7.11 createOamServer

Online and offline command that creates an OAM Server entry in the system configuration.

4.7.11.1 Description

Creates an OAM Server registration, including the host, port, registration name, OAM Proxy port and server ID, and, optionally, the OAM Proxy shared secret.

The scope of this command is an instance, only. The scope is not an argument

4.7.11.2 Syntax

```
createOamServer(host,port, paramNameValueList)
```

Argument	Definition
host	Mandatory. Specifies the name of the OAM Server host.
port	Mandatory. Specifies the listening port of the OAM Server host.
domainHome	Offline mode: Mandatory Online mode: Optional
paramNameValueList	Specifies the comma-separated list of parameter name-value pairs. The format of each pair is: `paramName='paramValue'` Mandatory: ■ `configurationProfile`—The name of this instance registration, which appears under Server Instances on the System Configuration tab in the OAM Administration Console. ■ `oamProxyPort`—The listening port of this instance. ■ `oamProxyServerID`—The name of the OAM Proxy for this server instance, which will appear under the OAM Proxy sub tab of the server instance in the OAM Administration Console. ■ `siteName`—siteName/serverName for the instance.

4.7.11.3 Example

The following invocation creates a configuration for `your_host` with listening port `15000`. The configuration entry in the Administration Console will be `oam_server1`. The OAM Proxy port is 3004 and the OAM Proxy Server ID is `AccessServerConfigProxy`:

```
createOamServer(host="my_host", port="15000", configurationProfile=
"oam_server1", oamProxyPort="3004", oamProxyServerID="ProxyID",
siteName="siteName1",domainHome="domainHome1")
```

4.7.12 editOamServer

Online and offline command that enables you to edit OAM Server registration details.

4.7.12.1 Description

Edits the registration for an OAM Server, which can include the host, port, registration name, OAM Proxy port and server ID, and, optionally, the OAM Proxy shared secret.

The scope of this command is an instance, only. The scope is not an argument.

4.7.12.2 Syntax

```
editOamServer(name, port, paramNameValueList)
```

Argument	Definition
name	Mandatory. Specifies the name of the OAM Server host.
port	Mandatory. Specifies the port number of the OAM Server host.
domainHome	Offline mode: Mandatory Online mode: Optional
paramNameValueList	Specifies the comma-separated list of parameter name-value pairs. The format of each pair is: `paramName='paramValue'` Mandatory: ■ `configurationProfile`—The name of this instance registration, which appears under Server Instances on the System Configuration tab in the OAM Administration Console. ■ `oamProxyPort`—The listening port of this instance. ■ `oamProxyServerID`—The name of the OAM Proxy for this server instance, which will appear under the OAM Proxy sub tab of the server instance in the OAM Administration Console. ■ `siteName`—siteName/serverName for the instance.

4.7.12.3 Example

You can use any of the optional attributes to change current settings. The following invocation enables you to add the OAM Proxy shared secret to the configuration entry `oam_server1`.

```
editOamServer(name="oam_server1", port="15000",configurationProfile=
"oam_server1", oamProxyPort="3004",oamProxyServerID="Proxy1",
siteName="siteName1",domainHome="domainHome1")
```

4.7.13 deleteOamServer

Online and offline command that enables you to delete the named OAM Server registration.

4.7.13.1 Description

Deletes an entire OAM Server configuration.

The scope of this command is an instance, only. The scope is not an argument.

4.7.13.2 Syntax

```
deleteOamServer(host,port)
```

Argument	Definition
host	Mandatory. Specifies the name of the OAM Server host.
port	Mandatory. Specifies the listening port of the OAM Server host.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.13.3 Example

The following invocation enables you to delete the OAM Server registration for `oam_server1` with listening port `15000`.

```
deleteOamServer(host="oam_server1",port="15000",domainHome="domainHome1")
```

4.7.14 displayOssoAgent

Online and offline command that displays OSSO Agent configuration details.

4.7.14.1 Description

Displays OSSO Agent registration details, which also appear in the OAM Administration Console.

The scope of this command is an instance, only. The scope is not an argument

4.7.14.2 Syntax

```
displayOssoAgent(agentName)
```

Argument	Definition
agentName	Mandatory. Specifies the name of the OSSO Agent.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.14.3 Example

The following invocation displays the OSSO Agent's registration information:

```
displayOssoAgent(agentName="OSSOAgent1",domainHome="domainHome1")
```

4.7.15 editOssoAgent

Online and offline command that enables you to edit an OSSO Agent registration.

4.7.15.1 Description

Changes OSSO Agent configuration details, including the Site Token, Success URL, Failure URL, Home URL, Logout URL, Start Date, End Date, Administrator ID, and Administrator Info.

The scope of this command is an instance, only. The scope is not an argument

4.7.15.2 Syntax

```
editOssoAgent(agentName,paramNameValueList)
```

Argument	Definition
agentName	Mandatory. Specifies the name of the OSSO Agent.
domainhome	Offline mode: Mandatory Online mode: Optional
paramNameValueList	Specifies the comma-separated list of parameter name-value pairs to be updated. The format of each pair is: `paramName='paramValue'` Optional: ■ `siteToken`—The Application Token used by the partner when requesting authentication. ■ `successUrl`—The redirect URL to be used upon successful authentication. ■ `failureUrl`—The redirect URL to be used if authentication fails. ■ `homeUrl`—The redirect URL to be used for the Home page after authentication. ■ `logoutUrl`—The redirect URL to be used when logging out. This redirects the user to the global logout page on the server ■ `startDate`—First month, day, and year for which login to the application is allowed by the server. ■ `endDate`—Final month, day, and year for which login to the application is allowed by the server. ■ `adminId`—Administrator login ID for this mod_osso instance. ■ `adminInfo`—Administrator details for this mod_osso instance.

4.7.15.3 Example

The following invocation changes the Administrator ID and information in the registration entry for `OSSOAgent1`:

```
editOssoAgent(agentName="OSSOAgent1", siteToken="siteToken",
successUrl="successUrl",failureUrl="failureUrl",homeUrl="homeUrl",
logoutUrl="logoutUrl",startDate="2009-12-10", endDate="2012-12-30",
adminId= 345", adminInfo="Agent11", domainHome="domainHome1")
```

4.7.16 deleteOssoAgent

Online and offline command that enables you to delete an OSSO Agent registration.

4.7.16.1 Description

Removes an OSSO Agent configuration.

The scope of this command is an instance, only. The scope is not an argument

4.7.16.2 Syntax

```
deleteOssoAgent(agentName)
```

Argument	Definition
agentName	Mandatory. Specifies the name of the OSSO Agent.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.16.3 Example

The following invocation removes the OSSO Agent named `OSSOAgent1`:

```
deleteOssoAgent(agentName="OSSOAgent1", domainHome="domainHome1")
```

4.7.17 displayWebgateAgent

Online and offline command that displays a 10g WebGate registration.

4.7.17.1 Description

Displays all 10*g* WebGate registration details, which can also be seen in the OAM Administration Console.

The scope of this command is an instance, only. The scope is not an argument

4.7.17.2 Syntax

```
displayWebgateAgent(agentName)
```

Argument	Definition
agentName	Mandatory. Specifies the name of the WebGate Agent.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.17.3 Example

The following invocation displays registration information for `my_WebGate`:

```
displayWebgateAgent(agentName=)"my_Webgate", domainHome="domainHome1")
```

4.7.18 editWebgateAgent

Online and offline command that enables you to edit a 10g WebGate registration.

4.7.18.1 Description

Enables you to change 10g WebGate Agent registration details.

The scope of this command is an instance, only. The scope is not an argument

4.7.18.2 Syntax

```
editWebgateAgent(agentName,paramNameValueList)
```

Argument	Definition
agentName	Mandatory. Specifies the name of the WebGate Agent.
domainhome	Offline mode: Mandatory Online mode: Optional

Argument	Definition
paramNameValueList	Specifies the comma-separated list of parameter name-value pairs to be updated. The format of each pair is: `paramName='paramValue'` Mandatory: ■ `agentId`—Name of the OAM Agent (WebGate). Optional: ■ `accessClientPassword`—An optional password for this WebGate Agent. ■ `state`—Whether the OAM Agent is enabled or disabled. ■ `preferredHost`—Prevents security holes that can be created if a host's identifier is not included in the Host Identifiers list. For virtual hosting, you must use the Host Identifiers feature. ■ `aaaTimeOutThreshold`—Number (in seconds) to wait for a response from the OAM Run-time Server. If this parameter is set, it is used as an application TCP/IP timeout instead of the default TCP/IP timeout. Default = -1 (default network TCP/IP timeout is used). ■ `security`—Level of transport security to and from the OAM Run-time Server: open, simple, or cert. ■ `primaryCookieDomain`—The Web server domain on which the OAM Agent is deployed, for instance,*.acompany.com*. ■ `maxConnections`—The maximum number of connections that this OAM Agent can establish with the OAM Server. This number must be the same as (or greater than) the number of connections that are actually associated with this agent. Default = 1. ■ `maxCacheElements`—Number of elements maintained in the cache. Cache elements are URLs or Authentication Schemes. The value of this setting refers to the maximum consolidated count for elements in both of these caches. Default = 10000. ■ `cacheTimeOut`—Amount of time cached information remains in the OAM Agent cache when the information is neither used nor referenced. Default = 1800 (seconds). ■ `cookieSessionTime`—Amount of time that the ObSSOCookie persists. Default = 3600 (seconds)*. ■ `maxSessionTime`—Maximum amount of time, in seconds, that a user's authentication session is valid regardless of their activity. At the expiration of this time, the user is re-challenged for authentication. This is a forced logout. Default = 3600 (seconds). A value of 0 disables this timeout setting. ■ `idleSessionTimeout`—Amount of time in seconds that a user's authentication session remains valid without accessing any OAM Agent protected resources. Default = 3600 (seconds). A value of 0 disables this timeout setting. ■ `failoverThreshold`—Number representing the point when this OAM Agent opens connections to a Secondary OAM Server. Default = 1.

4.7.18.3 Example

You can alter any or all of the settings. Use the following invocation to change specific information in the WebGate Agent registration, including the Agent ID, state, maximum connections, OAM Server timeout, primary cookie domain, cache time out,

cookie session timeout, maximum session timeout, idle session timeout, and failover threshold, as follows:

```
editWebgateAgent(agentName="my_WebGate", agentId="WebGate2", state=
"enabled", maxConnections="2", aaaTimeOutThreshold="2",
primaryCookieDomain="adomain.com", cacheTimeOut="1200",
cookieSessionTime=1500, maxSessionTime=1500, idleSessionTimeout=
"1500", failoverThreshold="25", domainHome="domainHome1")
```

4.7.19 deleteWebgateAgent

Online and offline command that enables you to delete a 10g WebGate Agent registration.

4.7.19.1 Description

Removes an 10*g* WebGate Agent registration.

The scope of this command is an instance, only. The scope is not an argument

4.7.19.2 Syntax

```
deleteWebgateAgent(agentName)
```

Argument	Definition
agentName	Mandatory. Specifies the name of the WebGate Agent.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.19.3 Example

The following invocation removes the WebGate Agent named `my_WebGate`:

```
deleteWebgateAgent(agentName="my_WebGate", domainHome="domainHome1")
```

4.7.20 changeLoggerSetting

Online and offline command that changes the logger level.

4.7.20.1 Description

Changes the level of one or more, or all, loggers.

The scope of this command is an instance, only. The scope is not an argument.

4.7.20.2 Syntax

```
changeLoggerSetting (loggerName='', loggerLevel=''):
```

Argument	Definition
loggerName	Optional. Specifies the OAM logger name. Multiple OAM logger names can be specified, separated by commas, or you can use the wildcard (*) character to specify all OAM collectors, which is the default.
loggerLevel	SEVERE, WARNING, INFO, CONFIG, FINE.

4.7.20.3 Example

The following invocation changes the logger level to SEVERE:

```
changeLoggerSetting(loggerName=" ", loggerLevel=SEVERE)
```

4.7.21 changeConfigDataEncryptionKey

Offline command that regenerates the configuration data encryption key.

4.7.21.1 Description

Regenerates the configuration data encryption key, re-encrypts the configuration data using the new key, and outputs attribute information of the identity store.

The scope of this command is an instance, only. The scope is not an argument.

4.7.21.2 Syntax

```
changePasswordEncKey (oldpassword='', newPassword='')
```

Argument	Definition
oldPassword	Mandatory. Specifies the password that retrieves the current encryption key.
newPassword	Mandatory. Defines a new password that protects the newly generated encryption key.

4.7.21.3 Example

The following invocation changes the old and new password, regenerates the key, and re-encrypts the configuration data:

```
changePasswordEncKey(oldpassword="oldpassword",
newPassword="newpassword")
```

4.7.22 displayUserIdentityStore

Online and offline command that displays user identity store registration information.

4.7.22.1 Description

Displays information of the user identity store registered with Oracle Access Manager.

The scope of this command is an instance, only. The scope is not an argument.

4.7.22.2 Syntax

```
displayUserIdentityStore(name)
```

Argument	Definition
name	Mandatory. Specifies the name of the LDAP user identity store.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.22.3 Example

The following invocation displays registration details of the user identity store:

```
displayUserIdentityStore(name="ID_store1", domainHome="domainHome1")
```

4.7.23 editUserIdentityStore

Online and offline command that changes attributes of the user identity store for Oracle Access Manager.

4.7.23.1 Description

Changes one or more attributes of the user identity store registered with Oracle Access Manager.

The scope of this command is an instance, only. The scope is not an argument.

4.7.23.2 Syntax

`editUserIdentityStore(name,paramNameValueList)`

Argument	Definition
name	Mandatory. Specifies the unique name of the LDAP user identity store (only upper and lower case alpha characters and numbers).
domainhome	Offline mode: Mandatory Online mode: Optional

paramNameValueList	Specifies the comma-separated list of parameter name-value pairs. The format of each pair is:
	`paramName='paramValue'`
	Include one or more of the following parameter name-value pairs, in addition to those in createUserIdentityStore, to change the OAM user identity store configuration:
	▪ `userFilterObjectClasses`—List of user filter object classes (separated by semi-colon).
	▪ `groupFilterObjectClasses`—List of group filter object classes (separated by semi-colon).
	▪ `referralPolicy`—LDAP referral policy (either "follow", "ignore" or "throw").
	▪ `searchTimeLimit`—Time limit in seconds for LDAP Search operation.
	▪ `minConnections`—Minimum number of connections in the connection pool.
	▪ `maxConnections`—Maximum number of connections in the connection pool.
	▪ `connectionWaitTimeout`—Number of seconds to wait for obtaining a connection from the pool.
	▪ `connectionRetryCount`—Number of attempts to establish a connection to identity store.
	▪ `groupNameAttr`—Name of the attribute to look up the user groups. For example:
	`ou=people,ou=myrealm,dc=base_domain`
	▪ `groupCacheEnabled`—Toggle (true/false) to enable LDAP group cache.
	▪ `groupCacheSize`—Number of entries in LDAP group cache.
	▪ `groupCacheTTL`—Total time to live for each entry of LDAP group cache.

4.7.23.3 Example

The following invocation changes the LDAP URL of the user identity store for OAM:

```
editUserIdentityStore(name="identity_store_name",
LDAP_url="ldap://localhost:7003", domainHome="domaonHome1")
```

4.7.24 createUserIdentityStore

Online and offline command that creates a user identity store registration for Oracle Access Manager.

4.7.24.1 Description

Creates an entry for a new user identity store to be registered with Oracle Access Manager.

The scope of this command is an instance, only. The scope is not an argument.

4.7.24.2 Syntax

```
createUserIdentityStore(name=,paramNameValueList)
```

Argument	Definition
name	Mandatory. Specifies the unique name of the LDAP user identity store (only upper and lower case alpha characters and numbers).
domainhome	Offline mode: Mandatory Online mode: Optional

Argument	Definition
paramNameValueList	Specifies the comma-separated list of parameter name-value pairs. The format of each pair is: `paramName='paramValue'` Mandatory: ■ `name`—The name for this user ID store. ■ `principal`—The login ID of the LDAP administrator. For example, cn=Admin. ■ `credential`—The password of the Principal, which is encrypted for security. ■ `type`—The type of the LDAP ID store to be created. ■ `userAttr`—User attributes of the store. ■ `usersearchbase`—The node under which user data is stored in the LDAP ID store to be created. For example: cn=users. ■ `groupSearchBase`—The node under which group data is stored in the LDAP ID store to be created. Mandatory Attribute. For example: cn=groups. ■ `ldapUrl`—The URL for the LDAP host, including port number of the LDAP ID store to be created. For example, `ldap://localhost:7001`. Optional: ■ `roleSecAdmin`—Name of the Admin group with all privileges for LDAP ID store. ■ `roleSysMonitor`—Name of the Admin group with read-only privileges for LDAP ID store to be created. ■ `roleSysManager`— Name of the Admin group with day-to-day operational privileges for LDAP ID store to be created. ■ `ldapProvider`—A supported LDAP provider. For example, OVD. ■ `isPrimary`—The designation of the primary User Identity Store. Boolean field. ■ `userIDProvider`—User Identity Provider of the store to be created. ■ `domainHome`—Domain Home location.

4.7.24.3 Example

The following invocation creates a new Oracle Internet Directory user identity store definition for use with Oracle Access Manager:

```
createUserIdentityStore(name="Name1",principal="Principal1",
credential="Credential1", type="OID", userAttr="userAttr1",
ldapProvider="ldapProvider", roleSecAdmin="roleSecAdmin1",
roleSysMonitor="roleSysMonitor", roleSysManager="roleSysManager",
roleAppAdmin="roleAppAdmin", userSearchBase="cn=users,
ldapUrl="ldapUrl", isPrimary="isPrimary", userIDProvider="userIDProvider",
groupSearchBase="cn=groups",domainHome="domainHome1")
```

4.7.25 deleteUserIdentityStore

Online and offline command that removes a Oracle Access Manager user identity store registration.

4.7.25.1 Description

Deletes the user identity store registered with Oracle Access Manager.

The scope of this command is an instance, only. The scope is not an argument.

4.7.25.2 Syntax

```
deleteUserIdentityStore(name)
```

Argument	Definition
name	Mandatory. Specifies the name of the LDAP user identity store to be removed.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.25.3 Example

The following invocation deletes the registration of the user identity store:

```
deleteUserIdentityStore(name="identity_store", domainHome="domainHome1")
```

4.7.26 configRequestCacheType

Online and offline command that configures the SSO server request cache type.

4.7.26.1 Description

Configures the SSO server request cache type.

The scope of this command is an instance, only. The scope is not an argument.

4.7.26.2 Syntax

```
configRequestCacheType(type)
```

Argument	Definition
type	Mandatory. Specifies `requestCacheType`. `requestCacheType`—The value of request cache type: BASIC or COOKIE.

4.7.26.3 Example

The following invocation identifies the request cache type as Cookie:

```
configRequestCacheType(type="COOKIE")
```

4.7.27 displayRequestCacheType

Online and offline command that displays the SSO server request cache type.

4.7.27.1 Description

Displays the SSO server request cache type entry.

The scope of this command is an instance, only. The scope is not an argument.

4.7.27.2 Syntax

```
displayRequestCacheType(domainHome)
```

Argument	Definition
type	Mandatory. Specifies requestCacheType. `requestCacheType`—The value of request cache type: BASIC or COOKIE.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.27.3 Example

The following invocation displays the request cache type.

```
displayRequestCacheType(domainHome="domainHome")
```

4.7.28 exportPolicy

Online only command that exports OAM policy data from a test (source) environment to the intermediate Oracle Access Manager file specified.

4.7.28.1 Description

Exports OAM policy data from a test (source) environment to the intermediate Oracle Access Manager file.

The scope of this command is an instance, only. The scope is not an argument.

4.7.28.2 Syntax

```
exportPolicy(pathTempOAMPolicyFile)
```

Argument	Definition
pathTempOAMPolicyFile	Mandatory. Specifies the path to the temporary Oracle Access Manager file.

4.7.28.3 Example

The following invocation specifies the path to the temporary file used when exporting policy data from a test (source) environment.

```
exportPolicy(pathTempOAMPolicyFile="oam_policy.xml")
```

4.7.29 importPolicy

Online only command that imports the OAM policy data from the intermediate Oracle Access Manager file specified.

4.7.29.1 Description

Imports the OAM policy data from the intermediate Oracle Access Manager file specified.

The scope of this command is an instance, only. The scope is not an argument.

4.7.29.2 Syntax

```
importPolicy(pathTempOAMPolicyFile)
```

Argument	Definition
pathTempOAMPolicyFile	Mandatory. Specifies the path to the temporary OAM file.

4.7.29.3 Example

The following invocation specifies the path to the temporary file used when importing policy data to a production (target).

```
importPolicy(pathTempOAMPolicyFile="oam_policy.xml")
```

4.7.30 importPolicyDelta

Online only command that imports the OAM policy changes from the intermediate Oracle Access Manager file specified.

4.7.30.1 Description

Imports the OAM policy changes from the intermediate Oracle Access Manager file specified.

The scope of this command is an instance, only. The scope is not an argument.

4.7.30.2 Syntax

```
importPolicyDelta(pathTempOAMPolicyFile)
```

Argument	Definition
pathTempOAMPolicyFile	Mandatory. Specifies the path to the temporary Oracle Access Manager file.

4.7.30.3 Example

The following invocation specifies the path to the temporary file used when importing only changed policy data to a production (target).

```
importPolicyDelta(pathTempOAMPolicyFile="oam_policy_delta.xml")
```

4.7.31 migratePartnersToProd

Online only command that migrates partners from the current (source) OAM Server to the specified (target) OAM Server.

4.7.31.1 Description

Migrates partners from the current (source) OAM Server to the specified (target) OAM Server.

The scope of this command is an instance, only. The scope is not an argument.

4.7.31.2 Syntax

```
migratePartnersToProd(prodServerHost,prodServerPort,prodServerAdminUser,prodServer
AdminPwd)
```

Argument	Definition
prodServerHost	Host name of the target OAM Server to which partners are to be migrated.

Argument	Definition
prodServerPort	Port of the target OAM Server to which partners are to be migrated.
prodServerAdminUser	Administrator of the target OAM Server to which partners are to be migrated.
prodServerAdminPwd	Target OAM Server administrator's password.

4.7.31.3 Example

The following invocation specifies the required information.

```
migratePartnersToProd(prodServerHost="host",prodServerPort="port",
prodServerAdminUser="weblogic",prodServerAdminPwd="welcome")
```

4.7.32 exportPartners

Online only command that exports Oracle Access Manager partners from the source to the intermediate Oracle Access Manager file specified.

4.7.32.1 Description

Exports the Oracle Access Manager partners from the source to the intermediate Oracle Access Manager file specified.

The scope of this command is an instance, only. The scope is not an argument.

4.7.32.2 Syntax

```
exportPartners(pathTempOAMPartnerFile)
```

Argument	Definition
pathTempOAMPartnerFile	Mandatory. Specifies the path to the temporary Oracle Access Manager partner file.

4.7.32.3 Example

The following invocation specifies the path to the intermediate OAM partners file.

```
exportPartners(pathTempOAMPartnerFile="oam_partners.xml")
```

4.7.33 importPartners

Online only command that imports Oracle Access Manager partners from the intermediate Oracle Access Manager file specified.

4.7.33.1 Description

Imports the OAM partners from the intermediate Oracle Access Manager file specified.

The scope of this command is an instance, only. The scope is not an argument.

4.7.33.2 Syntax

```
importPartners(pathTempOAMPartnerFile)
```

Argument	Definition
pathTempOAMPartnerFile	Mandatory. Specifies the path to the temporary OAM partner file.

4.7.33.3 Example

The following invocation specifies the path to the intermediate OAM partners file.

```
importPartners(pathTempOAMPartnerFile="oam_partners.xml")
```

4.7.34 configureOAAM

Online only command that configures the Oracle Access Manager-Oracle Adaptive Access Manager basic integration.

4.7.34.1 Description

Configures the OAM-OAAM basic integration.

The scope of this command is an instance, only. The scope is not an argument.

4.7.34.2 Syntax

```
configureOAAM(dataSourceName,paramNameValueList)
```

Argument	Definition
dataSourceName	Name of the data source to be created
paramNameValueList	Specifies the comma-separated list of parameter name-value pairs. The format of each pair is: `paramName='paramValue'` Mandatory: ■ `hostName`—The name of the database host. ■ `port`—Database port. ■ `sid`—The database sid (database identifier). ■ `userName`—OAAM schema name. ■ `passWord`—OAAM schema password. Optional: ■ `maxConnectionSize`—Max connection reserve time out size. ■ `maxPoolSize`—Maximum size for connection pool. ■ `serverName`—Target server for the data source.

4.7.34.3 Example

The following invocation configures the Oracle Access Manager-Oracle Adaptive Access Manager basic integration.

```
configureOAAM(dataSourceName = "MyOAAMDS", hostName = "host.us.co.com",
port = "1521", sid = "sid", userName = "username", passWord = "password",
maxConnectionSize = None, maxPoolSize = None, serverName = "oam_server1")
```

4.7.35 registerOIFDAPPartner

Online and offline command that registers Oracle Identity Federation as a Delegated Authentication Protocol (DAP) Partner.

4.7.35.1 Description

Registers Oracle Identity Federation as Delegated Authentication Protocol (DAP) Partner.

The scope of this command is an instance only. The scope is not an argument.

4.7.35.2 Syntax

```
registerOIFDAPPartner()
```

Argument	Definition
paramNameValueList	Specifies the comma-separated list of parameter name-value pairs. The format of each pair is: `paramName='paramValue'` Mandatory: Include the following parameter name-value pairs to create a new OAM user identity store configuration: ■ `keystoreLocation`—Location of the Keystore file (generated at the OIF Server.). ■ `logoutURL`—The OIF Server's logout URL. Optional: ■ `rolloverInterval`—The Rollover Interval for the keys used to encrypt/decrypt SASSO Tokens.

4.7.35.3 Example

The following invocation illustrates use of all parameters.

```
registerOIFDAPPartner(keystoreLocation="/scratch/keystore",
logoutURL="http://<oifhost>:<oifport>/fed/user/sploosso?doneURL=http://<oamhost>:
<oam port>/ngam/server/pages/logout.jsp", rolloverTime="526")
```

4.7.36 enableCoexistMode

Online command that enables the Coexist Mode.

4.7.36.1 Description

Enables the Coexist Mode.

The scope of this command is an instance, only. The scope is not an argument.

4.7.36.2 Syntax

```
enableCoexistMode()
```

4.7.36.3 Example

The following invocation enables the Coexist Mode.

```
enableCoexistMode
```

4.7.37 disableCoexistMode

Online command that disables the Coexist Mode.

4.7.37.1 Description

Disables the Coexist Mode.

The scope of this command is an instance, only. The scope is not an argument.

4.7.37.2 Syntax

```
disableCoexistMode()
```

4.7.37.3 Example

The following invocation enables the Coexist Mode.

```
disableCoexistMode
```

4.7.38 editGITOValues

Online and offline command that edits GITO configuration parameters.

4.7.38.1 Description

Edits GITO configuration parameters.

The scope of this command is an instance, only. The scope is not an argument.

4.7.38.2 Syntax

```
editGITOValues(gitoEnabled, paramNameValueList)
```

Argument	Definition
gitoEnabled	True (or false). Allows (or denies) user to set GITO enabled property.
domainhome	Offline mode: Mandatory Online mode: Optional
paramNameValueList	Specifies the comma-separated list of parameter name-value pairs. The format of each pair is: `paramName='paramValue'` Mandatory: Include the following parameter name-value pairs to create a new OAM user identity store configuration: ■ `gitoCookieDomain`—Allows user to set the GITO cookie domain entry. Optional: ■ `gitoCookieName`—Allows user to set the GITO cookie name. ■ `gitoVersion`—Allows user to set the GITO version. Can be ONLY v1.0 or v3.0. ■ `gitoTimeout`—Allows user to set the GITO timeout value. ■ `gitoSecureCookieEnabled`—True (or false). Allows (or denies) user to set the GITO cookie enabled property.

4.7.38.3 Example

The following invocation edits GITO configuration parameters.

```
editGITOValues(gitoEnabled="true",gitoCookieDomain=".abc.com",gitoCookieName="ABC"
,gitoVersion="v1.0",gitoTimeout="20",gitoSecureCookieEnabled="false",domainHome="/
```

```
abc/def/ijk")\n
```

4.7.39 editWebgate11gAgent

Online and offline command that edits an 11g WebGate registration.

4.7.39.1 Description

Edits an 11g WebGate registration.

The scope of this command is an instance, only. The scope is not an argument.

4.7.39.2 Syntax

```
editWebgate11gAgent(agentname, paramNameValueList)
```

Argument	Definition
`agentname`	Name of the registered OAM 11g WebGate agent to be edited.
`domainhome`	Offline mode: Mandatory Online mode: Optional

Argument	Definition

paramNameValueList	Specifies the comma-separated list of parameter name-value pairs. The format of each pair is:
	`paramName='paramValue'`
	Optional:
	`accessClientPassword`—Unique password for this WebGate
	`state`—Specifies whether or the agent is enabled or disabled.
	`security`—Level of communication security between the Agent and the OAM Server: Open, Simple, Cert.
	`aaaTimeOutThreshold`—Number (in seconds) to wait for a response from the OAM Server.
	`logOutUrls`—List of URLS that trigger the logout handler, which removes the ObSSOCookie.
	`maxConnections`—The maximum number of connections that this OAM Agent can establish with the OAM Server.
	`maxCacheElements`—Number of elements maintained in the cache.
	`cacheTimeout`—Amount of time cached information remains in the OAM Agent cache when the information is neither used nor referenced. Default = 1800 (seconds).
	`logoutCallbackUrl` —The URL to oam_logout_success, which clears cookies during the call back. By default, this is based on the Agent base URL supplied during agent registration. For example:
	`http://<host>:<port>`
	`maxSessionTime`—Maximum amount of time in seconds that a user's authentication session is valid, regardless of their activity.
	`logoutRedirectUrl`—The URL (absolute path) to the central logout page (logout.html). By default, this is based on the OAM Administration Console host name with a default port of 14200.
	`failoverThreshold`—Number representing the point when this OAM Agent opens connections to a Secondary OAM Server.
	`tokenValidityPeriod`—Amount of time in seconds that a user's authentication session remains valid without accessing any OAM Agent protected resources.
	`logoutTargetUrlParamName`—The value for this is name for the query parameter that the OPSS applications passes to WebGate during logout.

4.7.39.3 Example

The following invocation lists all mandatory and optional parameters.

```
editWebgate11gAgent(agentName="WebgateAgent1", accessClientPasswd = "welcome1",
state = "Enabled", preferredHost="141.144.168.148:2001", aaaTimeoutThreshold="10",
security = "open", logOutUrls = "http://<host>:<port>", maxConnections = "16"
maxCacheElems = "10000" , cacheTimeout = "1800", logoutCallbackUrl =
"http://<host>:<port>", maxSessionTime = "24", logoutRedirectUrl =
"logoutRedirectUrl", failoverThreshold = "1", tokenValidityPeriod="aPeriod"
logoutTargetUrlParamName = "logoutTargetUrl", domainHome="domainHome1")
```

4.7.40 deleteWebgate11gAgent

Online and offline command that enables you to delete an 11g WebGate Agent registration.

4.7.40.1 Description

Removes an 11*g* WebGate Agent registration.

The scope of this command is an instance, only. The scope is not an argument

4.7.40.2 Syntax

```
deleteWebgate11gAgent(agentName)
```

Argument	Definition
agentName	Mandatory. Specifies the name of the 11g WebGate Agent.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.40.3 Example

The following invocation removes the 11g WebGate Agent named `my_11gWebGate`:

```
deleteWebgate11gAgent(agentName="my_11gWebGate", domainHome="domainHome1")
```

4.7.41 displayWebgate11gAgent

Online and offline command that enables you to display an 11g WebGate Agent registration.

4.7.41.1 Description

Displays an 11*g* WebGate Agent registration.

The scope of this command is an instance, only. The scope is not an argument

4.7.41.2 Syntax

```
displayWebgate11gAgent(agentName)
```

Argument	Definition
agentName	Mandatory. Specifies the name of the WebGate Agent.
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.41.3 Example

The following invocation displays the WebGate Agent named `my_11gWebGate`:

```
displayWebgate11gAgent(agentName="my_11gWebGate", domainHome="domainHome1")
```

4.7.42 displayOAMMetrics

Online and offline command that enables the display of metrics of OAM Servers.

4.7.42.1 Description

Enables the display of metrics of OAM Servers.

The scope of this command is an instance, only. The scope is not an argument.

4.7.42.2 Syntax

```
displayOAMMetrics(domainHome)
```

Argument	Definition
domainhome	Offline mode: Mandatory Online mode: Optional

4.7.42.3 Example

The following invocation enables the display of metrics of OAM Servers.

```
displayOAMMetrics (domainHome=(domainHome1")
```

4.7.43 updateOIMHostPort

Online only command that updates the Oracle Identity Manager configuration when integrated with Oracle Access Manager.

4.7.43.1 Description

Updates the Oracle Identity manager configuration in system configuration.

The scope of this command is an instance, only. The scope is not an argument.

4.7.43.2 Syntax

```
updateOIMHostPort(hostname, port, secureProtocol)
```

Argument	Definition
hostname	Name of the Oracle Identity Manager host.
port	Port of the Oracle Identity Manager host.
secureProtocol	True or false.

4.7.43.3 Example

The following invocation illustrates this command.

```
updateOIMHostPort(hostName="OIM host", port="7777", secureProtocol="true")
```

4.7.44 configureOIM

Online only command that creates an agent registration specific to Oracle Identity Manager when integrated with Oracle Access Manager.

4.7.44.1 Description

Creates an Agent registration specific to Oracle Identity Manager when integrated with Oracle Access Manager.

The scope of this command is an instance, only. The scope is not an argument.

4.7.44.2 Syntax

```
updateOIMHostPort(hostname, port, secureProtocol)
```

Argument	Definition
hostname	Name of the Oracle Identity Manager host.
port	Port of the Oracle Identity Manager Managed Server.
oimSecureProtocolEnabled	True or false (depending on HTTP or HTTPS).
oimAccessGatePwd	If provided will be the agent password for Open mode
oimCookieDomain	Domain to which the cookie is to be set
oimWgId	Agent registration name.
oimWgVersion	Possible values 10g or 11g. If not provided, default is 10*g*.

4.7.44.3 Example

The following invocation illustrates this command.

```
updateOIMHostPort(hostName="OIM host", port="7777", secureProtocol="true")
configureOIM(oimHost="OIM host", oimPort="7777", oimSecureProtocolEnabled="true",
oimAccessGatePwd = "Access Gate Password", oimCookieDomain = "OIM Cookie Domain",
oimWgId="OIM Webgate ID", oimWgVersion="OIM Webgate Version")
```

4.7.45 updateOSSOResponseCookieConfig

Online and offline command that updates OSSO Proxy response cookie settings.

4.7.45.1 Description

Updates OSSO Proxy response cookie settings.

The scope of this command is an instance, only. The scope is not an argument.

4.7.45.2 Syntax

```
updateOSSOResponseCookieConfig()
```

Argument	Definition
domainhome	Offline mode: Mandatory Online mode: Optional
cookieName	Optional. Name of the cookie for which settings are updated. If not specified, the global setting is updated.
cookieMaxAge	Maximum age of a cookie in minutes. A negative value sets a session cookie.
isSecureCookie	Boolean flag specifies if cookie should be secure (sent only over SSL channel).
cookieDomain	The domain of the cookie.

4.7.45.3 Example

The following invocation illustrates this command.

```
updateOSSOResponseCookieConfig(cookieName = "<cookieName>",
cookieMaxAge = "<cookie age in minutes>", isSecureCookie = "true | false",
cookieDomain="<domain of the cookie>", domainHome = "<wls_domain_home_path>")
```

4.7.46 deleteOSSOResponseCookieConfig

Online and offline command that deletes OSSO Proxy response cookie settings.

4.7.46.1 Description

Deletes OSSO Proxy response cookie settings.

The scope of this command is an instance, only. The scope is not an argument.

4.7.46.2 Syntax

```
deleteOSSOResponseCookieConfig()
```

Argument	Definition
domainhome	Offline mode: Mandatory Online mode: Optional
cookieName	Mandatory. Name of the cookie for which settings are deleted.

4.7.46.3 Example

The following invocation illustrates this command.

```
deleteOSSOResponseCookieConfig(cookieName = "<cookieName>",
cookieDomain="<domain of the cookie>", domainHome = "<wls_domain_home_path>")
```

4.7.47 displaySimpleModeGlobalPassphrase

Displays the simple mode global passphrase in plain text from the system configuration.

4.7.47.1 Description

Online only command that displays the simple mode global passphrase in plain text from the system configuration.

4.7.47.2 Syntax

```
displaySimpleModeGlobalPassphrase
```

There are no arguments for this command.

4.7.47.3 Example

The following invocation illustrates this command.

```
displaySimpleModeGlobalPassphrase
```

4.7.48 exportSelectedPartners

Exports selected OAM Partners.

4.7.48.1 Description

Exports selected OAM Partners to the intermediate OAM file specified.

4.7.48.2 Syntax

exportSelectedPartners

Argument	Definition
pathTempOAMPartnerFile	The temporary file containing partners to be migrated.
partnersNameList	comma separated list of partner ids to be migrated

4.7.48.3 Example

The following invocation illustrates this command.

```
exportSelectedPartners (pathTempOAMPartnerFile="/exampleroot/parent/tempfile.extn"
partnersNameList="partner1,partner2"
```

4.7.49 migrateArtifacts

Migrates artifacts.

4.7.49.1 Description

Migrates artifacts based on the input artifact file.

4.7.49.2 Syntax

migrateArtifacts

Argument	Definition
path	Location of the artifacts file is present
password	Password used while generating original artifacts.
type	InPlace or OutOfPlace
isIncremental	true or false. If true, an incremental upgrade is done.

4.7.49.3 Example

The following invocation illustrates this command.

```
migrateArtifacts(path = "/exampleroot/parent/t", password = "password", type =
"InPlace", isIncremental="false")
```

4.7.50 registerThirdPartyTAPPartner

Registers any third party as a Trusted Authentication Protocol (TAP) Partner.

4.7.50.1 Description

Registers any third party as a Trusted Authentication Protocol (TAP) Partner.

4.7.50.2 Syntax

registerThirdPartyTAPPartner

Argument	Definition
path	Location of the artifacts file is present
password	Password used while generating original artifacts.

Argument	Definition
partnerName	Name of partner. Can be any name used for identifying the third party partner.
keystoreLocation	The jceks file location.
password	password
tapTokenVersion	Version of the Trusted Authentication Protocol.
tapScheme	Trusted Authentication Protocol Authn Scheme (TAPScheme, out of the box.)
tapRedirectUrl	Third party access URL.

4.7.50.3 Example

```
registerThirdPartyTAPPartner (partnerName="ThirdPartyTAPPartner",keystoreLocation=
"/scratch/DAPKeyStore/mykeystore.jks",password="test",tapTokenVersion="v2.0",
tapScheme="TAPScheme",tapRedirectUrl="http://thirdpartyserverhost:port/
loginPage.jsp");
```

4.8 Oracle Security Token Service

Table 4–7 describes the various types of WLST commands available for the Oracle Security Token Service.

Table 4–7 WLST Oracle Security Token Service Command Groups

OSTS Command Type	Description
Partner Commands	WLST commands related to tasks involving partners.
Relying Party Partner Mapping Commands	The WS Prefix to Relying Party Partner mappings are used to map a service URL, specified in the **AppliesTo** field of a WS-Trust RST request, to a partner of type Relying Party. The WS prefix string can be an exact service URL, or a URL with a parent path to the service URL. For example, if a mapping is defined to map a WS Prefix (http://test.com/service) to a Relying Party (RelyingPartyPartnerTest), then the following service URLs would be mapped to the Relying Party: http://test.com/service, http://test.com/service/calculatorService, http://test.com/service/shop/cart...
Partner Profiles Commands	WLST commands related to tasks involving partner profiles.
Issuance Templates Commands	WLST commands related to tasks involving issuance templates.
Validation Templates Commands	WLST commands related to tasks involving validation templates.

Use the WLST commands listed in Table 4–8 to manage Oracle Security Token Service

Table 4–8 WLST Commands Oracle Security Token Service

Use this command...	To...	Use with WLST...
Partner Commands		
getPartner	Retrieve a partner and print result.	Online
getAllRequesterPartners	Retrieve the names of Requester partners.	Online
getAllRelyingPartyPartners	Retrieve the names of all Relying Party partners.	Online
getAllIssuingAuthorityPartners	Retrieve the names of all Issuing Authority partners.	Online

Table 4–8 (Cont.) WLST Commands Oracle Security Token Service

Use this command...	To...	Use with WLST...
isPartnerPresent	Query OSTS to determine whether or not the partner exists in the Partner store.	Online
createPartner	Create a new Partner entry.	Online
updatePartner	Update an existing Partner entry based on the provided information.	Online
deletePartner	Delete a partner entry.	Online
getPartnerUsernameTokenUsername	Retrieve the partner's username value.	Online
getPartnerUsernameTokenPassword	Retrieve the partner's password value.	Online
setPartnerUsernameTokenCredential	Set the username and password values of a partner entry.	Online
deletePartnerUsernameTokenCredential	Remove the username and password values from a partner entry.	Online
getPartnerSigningCert	Retrieve the Base64 encoded signing certificate for the partner.	Online
getPartnerEncryptionCert	Retrieve the Base64 encoded encryption certificate for the partner.	Online
setPartnerSigningCert	Upload the signing certificate to the partner entry.	Online
setPartnerEncryptionCert	Upload the encryption certificate to the partner entry.	Online
deletePartnerSigningCert	Remove the signing certificate from the partner entry.	Online Offline
deletePartnerEncryptionCert	Remove the encryption certificate from the partner entry.	Online Offline
getPartnerAllIdentityAttributes	Retrieve and display all Identity mapping attributes used to map a token to a requester partner.	Online Offline
getPartnerIdentityAttribute	Retrieve and display the identity mapping attribute.	Online Offline
setPartnerIdentityAttribute	Set the identity mapping attribute for a requester partner.	Online Offline
deletePartnerIdentityAttribute	Delete the identity mapping attribute for a requester partner.	Online Offline
Relying Party Partner Mapping Commands		
getAllWSPrefixAndPartnerMappings	Retrieve and display all WS Prefixes.	Online Offline
getWSPrefixAndPartnerMapping	Retrieve and display the Relying Party Partner mapped to the specified wsprefix parameter.	Online Offline
createWSPrefixAndPartnerMapping	Create a new WS Prefix mapping to a Relying Partner.	Online Offline

Table 4–8 (Cont.) WLST Commands Oracle Security Token Service

Use this command...	To...	Use with WLST...
deleteWSPrefixAndPartnerMapping	Delete an existing WS Prefix mapping to a Relying Partner.	Online Offline
Partner Profiles Commands		
getAllPartnerProfiles	Retrieve the names of all the existing partner profiles.	Online
getPartnerProfile	Retrieve partner profile configuration data.	Online
createRequesterPartnerProfile	Create a new Requester Partner profile with default configuration data.	Online
createRelyingPartyPartnerProfile	Create a new Relying Party Partner profile with default configuration data.	Online
createIssuingAuthorityPartnerProfile	Create a new Issuing Authority Partner profile with default configuration data.	Online
deletePartnerProfile	Delete an existing partner profile.	Online
Issuance Template Commands		
getAllIssuanceTemplates	Retrieve the names of all the existing Issuance Templates.	Online Offline
getIssuanceTemplate	Retrieve configuration data of a specific Issuance Template.	Online
createIssuanceTemplate	Create a new Issuance Template with default configuration data.	Online
deleteIssuanceTemplate	Delete an existing Issuance Template.	Online Offline
Validation Template Commands		
getAllValidationTemplates	Retrieve the names of all the existing Validation Templates.	Online Offline
getValidationTemplate	Retrieve configuration data of a specific Validation Template.	Online Offline
createWSSValidationTemplate	Create a new WS Security Validation Template with default configuration data.	Online Offline
createWSTrustValidationTemplate	Create a new WS Trust Validation Template with default configuration data.	Online Offline
deleteValidationTemplate	Delete an existing Issuance Template.	Online Offline

4.8.1 getPartner

Online command that retrieves the Partner entry and prints out the configuration for this partner.

4.8.1.1 Description

Retrieves the Partner entry and prints out the configuration for this partner.

4.8.1.2 Syntax

```
getPartner(partnerId)
```

Argument	Definition
partnerId	Specifies the partnerId: the ID of the partner.

4.8.1.3 Example

The following invocation retrieves the Partner entry and prints out the configuration for `customPartner`:

```
getPartner(partnerId="customPartner")
```

4.8.2 getAllRequesterPartners

Online command that retrieves Requester type partners.

4.8.2.1 Description

Retrieves Requester type partners.

4.8.2.2 Syntax

```
getAllRequesterPartners()
```

4.8.2.3 Example

The following invocation retrieves Requester type partners:

```
getAllRequesterPartners()
```

4.8.3 getAllRelyingPartyPartners

Online command that retrieves Relying Party partners.

4.8.3.1 Description

Retrieves the Relying Party partners.

4.8.3.2 Syntax

```
getAllRelyingPartyPartners()
```

4.8.3.3 Example

The following invocation retrieves Relying Party partners:

```
getAllRelyingPartyPartners()
```

4.8.4 getAllIssuingAuthorityPartners

Online command that retrieves Issuing Authority partners and prints out the result.

4.8.4.1 Description

Retrieves the Issuing Authority partners and prints out the result.

4.8.4.2 Syntax

```
getAllIssuingAuthorityPartners()
```

4.8.4.3 Example

The following invocation retrieves Issuing Authority partners and prints out the result:

```
getAllIssuingAuthorityPartners()
```

4.8.5 isPartnerPresent

Online command that queries OSTS to determine whether or not the specified partner exists in the Partner store.

4.8.5.1 Description

Queries OSTS to determine whether or not the specified partner exists in the Partner store, and prints out the result.

4.8.5.2 Syntax

```
isPartnerPresent(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

4.8.5.3 Example

The following invocation queries OSTS to determine whether or not `customPartner` exists in the Partner store, and prints out the result:

```
isPartnerPresent(partnerId="customPartner")
```

4.8.6 createPartner

Online command that creates a new Partner entry.

4.8.6.1 Description

Creates a new Partner entry based on provided information. Displays a message indicating the result of the operation.

4.8.6.2 Syntax

```
createPartner(partnerId, partnerType, partnerProfileId, description,
bIsTrusted)
```

Argument	Definition
partnerId	Specifies the ID of the new partner to be created.
partnerType	Specifies the type of partner. Values can be one of the following: ■ STS_REQUESTER for Requester ■ STS_RELYING_PARTY for Relying Party ■ STS_ISSUING_AUTHORITY for Issuing Authority
partnerProfileId	Specifies the profile ID to be attached to this partner. It must reference an existing partner profile, and the type of the partner profile must be compliant with the type of the new partner entry.
description	Specifies the optional description of this new partner entry.

Argument	Definition
bIsTrusted	A value that indicates whether or not this new partner is trusted. Value can be either: ■ true for trusted ■ false if not trusted

4.8.6.3 Example

The following invocation creates STS_Requestor partner, `customPartner`, `custom-partnerprofile` with a description (`custom requester`), with a trust value of `true`, displays a message indicating the result of the operation:

```
createPartner(partnerId="customPartner", partnerType="STS_REQUESTER",
partnerProfileId="custom-partnerprofile", description="custom requester",
bIsTrusted="true")
```

4.8.7 updatePartner

Online command that updates an existing Partner entry.

4.8.7.1 Description

Updates an existing Partner entry based on the provided information. Displays a message indicating the result of the operation.

4.8.7.2 Syntax

```
updatePartner(partnerId, partnerProfileId, description, bIsTrusted)
```

Argument	Definition
partnerId	Specifies the ID of the new partner to be updated.
partnerProfileId	Specifies the partner profile ID. It must reference an existing partner profile, and the type of the partner profile must be compliant with the type of the new partner entry.
description	Specifies the optional description f this new partner entry.
bIsTrusted	A value that indicates whether or not this new partner is trusted. Value can be either: ■ true for trusted ■ false if not trusted

4.8.7.3 Example

The following invocation updates `customPartner` with a new profile ID, (`x509-wss-validtemp`), description (`custom requester with new profile id`), and a trust value of `false`. A message indicates the result of the operation:

```
updatePartner(partnerId="customPartner", partnerProfileId="x509-wss-validtemp",
description="custom requester with new profile id", bIsTrusted="false")
```

4.8.8 deletePartner

Online command that deletes a partner entry from OSTS.

4.8.8.1 Description

Deletes an existing Partner entry referenced by the `partnerId` parameter from OSTS, and prints out the result of the operation.

4.8.8.2 Syntax

```
deletePartner(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner to be deleted.

4.8.8.3 Example

The following invocation deletes the `customPartner` partner entry referenced by the partnerId parameter from OSTS, and prints out the result of the operation:

```
deletePartner(partnerId="customPartner")
```

4.8.9 getPartnerUsernameTokenUsername

Online command that retrieves a partner's username value that will be used for UNT credentials partner validation or mapping operation.

4.8.9.1 Description

Retrieves a partner's username value that will be used for UNT credentials partner validation or mapping operation, and displays the value.

4.8.9.2 Syntax

```
getPartnerUsernameTokenUsername(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

4.8.9.3 Example

The following invocation retrieves the `customPartner` partner username value that will be used for UNT credentials partner validation or mapping operation, and displays the value:

```
getPartnerUsernameTokenUsername(partnerId="customPartner")
```

4.8.10 getPartnerUsernameTokenPassword

Online command that retrieves a partner's password value that will be used for UNT credentials partner validation or mapping operation.

4.8.10.1 Description

Retrieves a partner password value that will be used for UNT credentials partner validation or mapping operation, and displays the value.

4.8.10.2 Syntax

```
getPartnerUsernameTokenPassword(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

4.8.10.3 Example

The following invocation retrieves `customPartner` partner password value that will be used for UNT credentials partner validation or mapping operation, and displays the value:

```
getPartnerUsernameTokenPassword(partnerId="customPartner")
```

4.8.11 setPartnerUsernameTokenCredential

Online command that sets the username and password values of a partner entry, that will be used for UNT credentials partner validation or mapping operation.

4.8.11.1 Description

Sets the username and password values of a partner entry, that will be used for UNT credentials partner validation or mapping operation. Displays the result of the operation.

4.8.11.2 Syntax

```
setPartnerUsernameTokenCredential(partnerId, UTUsername, UTPassword)
```

Argument	Definition
partnerId	Specifies the ID of the partner.
UTUsername	Specifies the username value used for UNT credentials validation or mapping operations.
UTPassword	Specifies the username value used for UNT credentials validation or mapping operations.

4.8.11.3 Example

The following invocation sets the username and password values of the `customPartner` partner entry, and displays the result of the operation:

```
setPartnerUsernameTokenCredential(partnerId="customPartner", UTUsername="test",
UTPassword="password")
```

4.8.12 deletePartnerUsernameTokenCredential

Online command that removes the username and password values from a partner entry that are used for UNT credentials partner validation or mapping operation, and displays the result of the operation.

4.8.12.1 Description

Removes the username and password values from a partner entry that are used for UNT credentials partner validation or mapping operation, and displays the result of the operation.

4.8.12.2 Syntax

```
deletePartnerUsernameTokenCredential(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner to be deleted.

4.8.12.3 Example

The following invocation removes the username and password values from a partner entry that are used for UNT credentials partner validation or mapping operation, and displays the result of the operation:

```
deletePartnerUsernameTokenCredential(partnerId="customPartner")
```

4.8.13 getPartnerSigningCert

Online command that retrieves the Base64 encoded signing certificate for the partner referenced by the partnerId parameter, and displays its value, as a Base64 encoded string.

4.8.13.1 Description

Retrieves the Base64 encoded signing certificate for the partner referenced by the partnerId parameter, and displays its value, as a Base64 encoded string.

4.8.13.2 Syntax

```
getPartnerSigningCert(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

4.8.13.3 Example

The following invocation retrieves Base64 encoded signing certificate for the partner referenced by the partnerId parameter, and displays its value, as a Base64 encoded string:

```
getPartnerSigningCert(partnerId="customPartner")
```

4.8.14 getPartnerEncryptionCert

Online command that retrieves the Base64 encoded encryption certificate, and displays its value as a Base64 encoded string.

4.8.14.1 Description

Retrieves the Base64 encoded encryption certificate for the partner referenced by the partnerId parameter, and displays its value as a Base64 encoded string.

4.8.14.2 Syntax

```
getPartnerEncryptionCert(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

4.8.14.3 Example

The following invocation retrieves the Base64 encoded encryption certificate for the partner referenced by the partnerId parameter, and displays its value, as a Base64 encoded string:

```
getPartnerEncryptionCert(partnerId="customPartner")
```

4.8.15 setPartnerSigningCert

Online command that Uploads the provided certificate to the partner entry as the signing certificate. Displays the result of the operation.

4.8.15.1 Description

Uploads the provided certificate to the partner entry (referenced by the partnerId parameter) as the signing certificate. The supported formats of the certificate are DER and PEM. Displays the result of the operation.

4.8.15.2 Syntax

```
setPartnerSigningCert(partnerId, certFile)
```

Argument	Definition
partnerId	Specifies the ID of the partner.
certFile	Specifies the location of the certificate on the local filesystem. Supported formats of the certificate are DER and PEM.

4.8.15.3 Example

The following invocation uploads the provided certificate to the partner entry `customPartner` as the signing certificate. Displays the result of the operation:

```
setPartnerSigningCert(partnerId="customPartner", certFile="/temp/signing_cert")
```

4.8.16 setPartnerEncryptionCert

Online command that Uploads the provided certificate to the partner entry as the encryption certificate. Displays the result of the operation.

4.8.16.1 Description

Uploads the provided certificate to the partner entry (referenced by the partnerId parameter) as the encryption certificate. Displays the result of the operation.

4.8.16.2 Syntax

```
setPartnerEncryptionCert(partnerId, certFile)
```

Argument	Definition
partnerId	Specifies the ID of the partner.
certFile	Specifies the location of the certificate on the local filesystem. Supported formats of the certificate are DER and PEM.

4.8.16.3 Example

The following invocation uploads the provided certificate to the partner entry `customPartner` as the signing certificate. Displays the result of the operation:

```
setPartnerSigningCert(partnerId="customPartner", certFile="/temp/signing_cert")
```

4.8.17 deletePartnerSigningCert

Online command that removes the encryption certificate from the partner entry and displays the result of the operation.

4.8.17.1 Description

Removes the encryption certificate from the partner entry, referenced by the partnerId parameter, and displays the result of the operation.

4.8.17.2 Syntax

```
deletePartnerSigningCert(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

4.8.17.3 Example

The following invocation removes the encryption certificate from the partner entry, `customPartner`, and displays the result of the operation:

```
deletePartnerSigningCert(partnerId="customPartner")
```

4.8.18 deletePartnerEncryptionCert

Online command that removes the signing certificate from the partner entry and displays the result of the operation.

4.8.18.1 Description

Removes the signing certificate from the partner entry, referenced by the partnerId parameter, and displays the result of the operation.

4.8.18.2 Syntax

```
deletePartnerEncryptionCert(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

4.8.18.3 Example

The following invocation removes the signing certificate from the partner entry, `customPartner`, and displays the result of the operation:

```
deletePartnerEncryptionCert(partnerId="customPartner")
```

4.8.19 getPartnerAllIdentityAttributes

Online command that retrieves and displays all the identity mapping attributes used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner.

4.8.19.1 Description

Retrieves and displays all the identity mapping attributes used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner.

The identity mapping attributes only exist for partners of type Requester.

4.8.19.2 Syntax

```
getPartnerAllIdentityAttributes(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the Requester partner. Identity mapping attributes only exist for partners of type Requester

4.8.19.3 Example

The following invocation retrieves and displays all the identity mapping attributes used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner: `customPartner`.

```
getPartnerAllIdentityAttributes(partnerId="customPartner")
```

4.8.20 getPartnerIdentityAttribute

Online command that retrieves and displays identity mapping attributes used to map a token or to map binding data to a requester partner.

4.8.20.1 Description

Retrieves and displays an identity mapping attribute used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner.

The identity mapping attributes only exist for partners of type Requester.

4.8.20.2 Syntax

```
getPartnerIdentityAttribute(partnerId, identityAttributeName)
```

Argument	Definition
partnerId	Specifies the ID of the Requester partner.
IdentityAttributeName	Specifies the name of the identity mapping attribute to retrieve and display. For example: `httpbasicusername`.

4.8.20.3 Example

The following invocation retrieves and displays one `identityAttribute` and its value as specified by `identityAttributeName`.

```
getPartnerIdentityAttribute(partnerId="customPartner",
identityAttributeName="httpbasicusername")
```

4.8.21 setPartnerIdentityAttribute

Online command that sets the identity mapping attribute for the Requester partner.

4.8.21.1 Description

Set the identity mapping attribute specified by `identityAttributeName` for the partner of type requester specified by the partnerId parameter. These identity mapping attributes only exist for Requester partners. Displays the result of the operation.

4.8.21.2 Syntax

```
setPartnerIdentityAttribute(partnerId, identityAttributeName,
identityAttributeValue)
```

Argument	Definition
partnerId	Specifies the ID of the partner of type Requester.
identityAttributeName	Specifies the name of the identity mapping attribute to retrieve and display.
identityAttributeValue	Specifies the value of the identity mapping attribute to set.

4.8.21.3 Example

The following invocation sets the identity mapping attribute specified by `identityAttributeName` for the Requester partner of type requester specified by the partnerId parameter. Displays the result of the operation.

```
setPartnerIdentityAttribute(partnerId="customPartner",
identityAttributeName="httpbasicusername",identityAttributeValue="test")
```

4.8.22 deletePartnerIdentityAttribute

Online command that deletes the identity mapping attribute.

4.8.22.1 Description

Deletes the identity mapping attribute specified by `identityAttributeName`.

The identity mapping attributes used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner, and they only exist for Requester partners.

4.8.22.2 Syntax

```
deletePartnerIdentityAttribute(partnerId, identityAttributeName)
```

Argument	Definition
partnerId	Specifies the ID of the partner.
identityAttributeName	Specifies the name of the identity mapping attribute to delete.

4.8.22.3 Example

The following invocation deletes the identity mapping attribute specified by `identityAttributeName` for Requester partner `customPartner`.

```
deletePartnerIdentityAttribute(partnerId="customPartner",
identityAttributeName="httpbasicusername")
```

4.8.23 getAllWSPrefixAndPartnerMappings

Online command that retrieves and displays all WS Prefixes to Relying Party Partner mappings.

4.8.23.1 Description

Retrieves and displays all WS Prefixes to Relying Party Partner mappings.

4.8.23.2 Syntax

```
getAllWSPrefixAndPartnerMappings()
```

4.8.23.3 Example

The following invocation retrieves and displays theWS Prefixes.

```
getAllWSPrefixAndPartnerMappings()
```

4.8.24 getWSPrefixAndPartnerMapping

Online command that retrieves and displays the Relying Party Partner mapped to the specified wsprefix parameter, if a mapping for that WS Prefix exists.

4.8.24.1 Description

Retrieves and displays the Relying Party Partner mapped to the specified wsprefix parameter, if a mapping for that WS Prefix exists.

4.8.24.2 Syntax

```
getWSPrefixAndPartnerMapping(wsprefix)
```

Argument	Definition
wsprefix	Specifies the WS Prefix entry to retrieve and display. The path is optional. If specified, it should take the following form: *http_protocol://hostname_ip/path*

4.8.24.3 Example

The following invocation retrieves nd displays the Relying Party Partner mapped to the specified wsprefix parameter, if a mapping for that WS Prefix exists.

```
getWSPrefixAndPartnerMapping(wsprefix="http://host1.example.com/path")
```

4.8.25 createWSPrefixAndPartnerMapping

Online command that creates a new WS Prefix mapping to a Relying Partner.

4.8.25.1 Description

Creates a new WS Prefix mapping to a Relying Partner referenced by the partnerid parameter, and displays the result of the operation.

4.8.25.2 Syntax

```
createWSPrefixAndPartnerMapping(wsprefix, partnerid, description)
```

Argument	Definition
wsprefix	Specifies the WS Prefix entry to retrieve and display. The path is optional. If specified, it should take the following form: *http_protocol://hostname_ip/path*
partnerId	Specifies the ID of the partner.
description	Specifies an optional description.

4.8.25.3 Example

The following invocation creates a new WS Prefix mapping to a Relying Partner Partner referenced by the partnerid parameter, and displays the result of the operation.

```
createWSPrefixAndPartnerMapping(wsprefix="http://host1.example.com/path",
partnerid="customRPpartner", description="some description")
```

4.8.26 deleteWSPrefixAndPartnerMapping

Online command that deletes an existing mapping of WS Prefix to a Relying Partner Partner.

4.8.26.1 Description

Deletes an existing mapping of WS Prefix to a Relying Partner, and displays the result of the operation.

4.8.26.2 Syntax

```
deleteWSPrefixAndPartnerMapping(wsprefix)
```

Argument	Definition
wsprefix	Specifies the WS Prefix entry to retrieve and display. The path is optional. If specified, it should take the following form: *http_protocol://hostname_ip/path*

4.8.26.3 Example

The following invocation deletes the existing mapping of WS Prefix to a Relying Partner, and displays the result of the operation.

```
deleteWSPrefixAndPartnerMapping(wsprefix="http://host1.example.com/path")
```

4.8.27 getAllPartnerProfiles

Online command that retrieves the names of all the existing partner profiles and displays them.

4.8.27.1 Description

Retrieves the names of all the existing partner profiles and displays them.

4.8.27.2 Syntax

```
getAllPartnerProfiles()
```

4.8.27.3 Example

The following invocation retrieves the names of all the existing partner profiles and displays them.

```
getAllPartnerProfiles()
```

4.8.28 getPartnerProfile

Online command that retrieves the configuration data of a specific partner profile, and displays the content of the profile.

4.8.28.1 Description

Retrieves the configuration data of the partner profile referenced by the partnerProfileId parameter, and displays the content of the profile.

4.8.28.2 Syntax

```
getPartnerProfile(partnerProfileId)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile.

4.8.28.3 Example

The following invocation retrieves the configuration data of the partner profile referenced by the partnerProfileId parameter, and displays the content of the profile.

```
getPartnerProfile(partnerProfileId="custom-partnerprofile")
```

4.8.29 createRequesterPartnerProfile

Online command that creates a new requester partner profile with default configuration data.

4.8.29.1 Description

Creates a new requester partner profile with default configuration data, and displays the result of the operation.

Table 4–9 describes the default configuration created with this command.

Table 4–9 Default Configuration: createRequesterPartnerProfile

Element	Description
Return Error for Missing Claims	Default: false
Allow Unmapped Claims	Default: false

Table 4–9 (Cont.) Default Configuration: createRequesterPartnerProfile

Element	Description

Token Type Configuration	The Token Type Configuration table includes the following entries. There are no mappings of token type to WS-Trust Validation Template:
	■ SAML 1.1 token type mapped to the following External URI:
	`http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1`
	The SAML 1.1 token type is not mapped to any WS-Trust Validation Template.
	■ SAML 2.0 token type mapped to the following External URI:
	`http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0`
	The SAML 2.0 token type is not mapped to any WS-Trust Validation Template.
	■ Username token type mapped to the following External URI:
	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken`
	The Username token type is not mapped to any WS-Trust Validation Template.
	Note: Token Type Configuration and token type to Validation Template mapping are both empty

Attribute Name Mapping	Default: The Attribute Name Mapping table is empty by default.

4.8.29.2 Syntax

```
createRequesterPartnerProfile(partnerProfileId, defaultRelyingPartyPPID,
description)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile.
defaultRelyingPartyPPID	Specifies the relying party partner profile to use, if the AppliesTo field is missing from the RST or if it could not be mapped to a Relying Party Partner.
description	Specifies the optional description for this partner profile

4.8.29.3 Example

The following invocation creates a new requester partner profile with default configuration data, and displays the result of the operation. For default data descriptions, see Table 4–9.

```
createRequesterPartnerProfile(partnerProfileId="custom-partnerprofile",
defaultRelyingPartyPPID="rpPartnerProfileTest", description="custom
partner profile")
```

4.8.30 createRelyingPartyPartnerProfile

Online command that creates a new relying party partner profile with default configuration data.

4.8.30.1 Description

Creates a new relying party partner profile with default configuration data, and displays the result of the operation.

Table 4–10 describes the default configuration created with this command.

Table 4–10 Default Configuration: createRelyingPartyPartnerProfile

Element	Description
Download Policy	Default: false
Allow Unmapped Claims	Default: false
Token Type Configuration	The Token Type Configuration will contain a single entry, with: ■ The token type set to the type of Issuance Template referenced by defaultIssuanceTemplateID ■ The Issuance template set to defaultIssuanceTemplateID **Note**: For the token type of the issuance template referenced by defaultIssuanceTemplateID, it will be linked to the issuance template, while the other token types will not be linked to any issuance template. If the issuance template referenced by defaultIssuanceTemplateID is of custom token type, the table will only contain one entry, with the custom token type, mapped to the custom token type as the external URI, and mapped to the issuance template referenced by defaultIssuanceTemplateID
Attribute Name Mapping	The Attribute Name Mapping table is empty be default.

4.8.30.2 Syntax

```
createRelyingPartyPartnerProfile(partnerProfileId, defaultIssuanceTemplateID,
description)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile.
defaultIssuanceTemplateID	Specifies the default issuance template and token type to issue if no token type was specified in the RST.
description	Specifies the optional description for this partner profile

4.8.30.3 Example

The following invocation creates a new relying party partner profile with default configuration data, and displays the result of the operation.

```
createRelyingPartyPartnerProfile(partnerProfileId="custom-partnerprofile",
defaultIssuanceTemplateID="saml11-issuance-template", description="custom partner
profile")
```

4.8.31 createIssuingAuthorityPartnerProfile

Online command that creates a new issuing authority partner profile with default configuration data.

4.8.31.1 Description

Creates a new issuing authority partner profile with the default configuration data in Table 4–11, and displays the result of the operation.

Table 4–11 Default Configuration: createIssuingAuthorityPartnerProfile

Element	Description
Server Clockdrift	Default: 600 seconds
Token Mapping	The Token Mapping Section will be configured as follows: ■ Override Simple User Mapping: false ■ Override User NameID Mapping: false ■ Override Attribute Based User Mapping: false ■ Override Simple Partner Mapping: false ■ Override Partner NameID Mapping: false Empty fields ■ simple user mapping ■ attribute based user mapping ■ simple partner mapping
Partner NameID Mapping	The Partner NameID Mapping table will be provisioned with the following entries as NameID format. However, without any data in the datastore column the issuance template referenced by defaultIssuanceTemplateID is of token type SAML 1.1, SAML 2.0, or Username. The table will contain the following entries: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomain QualifiedName ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
User NameID Mapping	The User NameID Mapping table will be provisioned with the following entries as NameID format: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomain QualifiedName, empty datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName, dn set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress, mail set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent, empty datastore column

Table 4–11 (Cont.) Default Configuration: createIssuingAuthorityPartnerProfile

Element	Description
Attribute Mapping	The Attribute Value Mapping and Attribute Name Mapping table is empty be default.

4.8.31.2 Syntax

```
createIssuingAuthorityPartnerProfile(partnerProfileId, description)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile.
description	Specifies the optional description for this partner profile

4.8.31.3 Example

The following invocation a new issuing authority partner profile with default configuration data, and displays the result of the operation.

```
createIssuingAuthorityPartnerProfile(partnerProfileId="custom-partnerprofile"
description="custom partner profile")
```

4.8.32 deletePartnerProfile

Online command that deletes an partner profile referenced by the partnerProfileId parameter.

4.8.32.1 Description

Deletes an partner profile referenced by the partnerProfileId parameter, and displays the result of the operation.

4.8.32.2 Syntax

```
deletePartnerProfile(partnerProfileId)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile to be removed.

4.8.32.3 Example

The following invocation deletes an partner profile referenced by the partnerProfileId parameter, and displays the result of the operation.

```
deletePartnerProfile(partnerProfileId="custom-partnerprofile")
```

4.8.33 getAllIssuanceTemplates

Online command that retrieves the names of all the existing issuance templates.

4.8.33.1 Description

Retrieves the names of all the existing issuance templates and displays them.

4.8.33.2 Syntax

```
getAllIssuanceTemplates
```

4.8.33.3 Example

The following invocation retrieves the names of all the existing issuance templates and displays them.

```
getAllIssuanceTemplates
```

4.8.34 getIssuanceTemplate

Online command that retrieves the configuration data of a specific issuance template.

4.8.34.1 Description

Retrieves the configuration data of the issuance template referenced by the issuanceTemplateId parameter, and displays the content of the template.

4.8.34.2 Syntax

```
getIssuanceTemplate(issuanceTemplateId)
```

Argument	Definition
issuanceTemplateId	Specifies the name of the issuance template.

4.8.34.3 Example

The following invocation retrieves the configuration data of the issuance template referenced by the issuanceTemplateId parameter, and displays the content of the template.

```
getIssuanceTemplate(issuanceTemplateId="custom-issuancetemp")
```

4.8.35 createIssuanceTemplate

Online command that creates a new issuance template with default configuration data.

4.8.35.1 Description

Creates a new issuance template with default configuration data, and displays the result of the operation.

Table 4–12 describes the default configuration for this command.

Table 4–12 Default Configuration: createIssuanceTemplate

Token Type	Description
Username	The issuance template will be created with the following default values: ■ Send Encrypted Token: false ■ NameID User Attribute: uid ■ NameID User Attribute Store: User Store ■ Password Attribute: (empty) ■ Include Nonce: true ■ Include Timestamp: true

Table 4–12 (Cont.) Default Configuration: createIssuanceTemplate

Token Type	Description
SAML 1.1 or SAML 2.0	The issuance template will be created with the following default values: ■ Send Encrypted Token: false ■ Assertion Issuer: OAM Hostname ■ NameID Format: Email Address ■ NameID User Attribute: mail ■ NameID User Attribute Store: User Store ■ NameID Qualifier: (empty) ■ Include Authn Statement: true ■ Include Attr Statement: true ■ Sign Assertion: true ■ Include Certificate in Signature: true ■ Send Encrypted NameID: false (SAML 2.0 only) ■ Default Subject Confirmation Method: Sender Vouches ■ Compute HOK Symmetric Key: true ■ HOK Symmetric Key Generation Algorithm: `http://www.w3.org/2001/04/xmlenc#aes128-cbc` Empty tables: Attribute Name Mapping, Attribute Value Mapping and Attribute Value Filter
Custom Type	The issuance template will be created with the following default values: ■ Send Encrypted Token: false

4.8.35.2 Syntax

```
createIssuanceTemplate(issuanceTemplateId, tokenType, signingKeyId,
description)
```

Argument	Definition
issuanceTemplateId	Specifies the name of the issuance template to be created.
tokenType	Possible values can be: ■ username: indicates that the token type is UsernameToken ■ saml11: indicates that the token type is a SAML 1.1 Assertion ■ saml20: indicates that the token type is a SAML 2.0 Assertion ■ <other>: in this case, the token type is assumed to be a custom token type, referenced by <other> (replace <other> by a value)
signingKeyId	Specifies the keyID referencing the key entry (defined in the STS General Settings UI section) that will be used to sign outgoing SAML Assertions. Only required when token type is saml11 or saml20.
description	An optional description.

4.8.35.3 Example

The following invocation creates a new issuance template with default configuration data, and displays the result of the operation.

```
createIssuanceTemplate(issuanceTemplateId="custom-issuancetemp",
```

```
tokenType="saml20", signingKeyId="osts_signing", description="custom issuance
template")
```

4.8.36 deleteIssuanceTemplate

Online command that deletes an issuance template referenced by the issuanceTemplateId parameter, and displays the result of the operation.

4.8.36.1 Description

Deletes an issuance template referenced by the issuanceTemplateId parameter, and displays the result of the operation.

4.8.36.2 Syntax

```
deleteIssuanceTemplate(issuanceTemplateId)
```

Argument	Definition
issuanceTemplateId	Specifies the name of the existing issuance template to be removed.

4.8.36.3 Example

The following invocation deletes an issuance template referenced by the issuanceTemplateId parameter, and displays the result of the operation.

```
deleteIssuanceTemplate(issuanceTemplateId="custom-issuancetemp")
```

4.8.37 getAllValidationTemplates

Online command that retrieves the names of all the existing validation templates.

4.8.37.1 Description

Retrieves the names of all the existing validation templates and displays them.

4.8.37.2 Syntax

```
getAllValidationTemplates()
```

4.8.37.3 Example

The following invocation retrieves the names of all the existing validation templates and displays them.

```
getAllValidationTemplates()
```

4.8.38 getValidationTemplate

Online command that retrieves the configuration data of a specific validation template, and displays the content of the template.

4.8.38.1 Description

Retrieves the configuration data of the validation template referenced by the validationTemplateId parameter, and displays the content of the template.

4.8.38.2 Syntax

```
getValidationTemplate(validationTemplateId)
```

Argument	Definition
validationTemplateId	Specifies the name of the existing validation template.

4.8.38.3 Example

The following invocation retrieves the configuration data of a specific validation template, and displays the content of the template.

```
getValidationTemplate(validationTemplateId="custom-wss-validtemp")
```

4.8.39 createWSSValidationTemplate

Online command that creates a new validation template with default configuration data.

4.8.39.1 Description

Creates a new validation template with default configuration data, and displays the result of the operation.

The WSS validation template is created with the values in Table 4–13, depending on the token type.

Table 4–13 Default Configuration: createWSSValidationTemplate

Token Type	Description
Username	The validation template will be created with the following default values: ■ Timestamp Lifespan: 600 seconds ■ Enable Credential Validation: true ■ Validation Source: Partner ■ Token Mapping: Map token to Partner ■ Enable Simple Partner Mapping: true ■ Partner Datastore Attribute: username

Table 4–13 (Cont.) Default Configuration: createWSSValidationTemplate

Token Type	Description
SAML 1.1 or SAML 2.0	The validation template will be created with the following default values: ■ Authentication Timeout: 3600 seconds ■ Timestamp Lifespan: 3600 seconds The Token Mapping section will be created with the following default values: ■ Map token: Map token to Partner ■ Enable Simple User Mapping: false ■ Enable User NameID Mapping: false ■ Enable Attribute Based User Mapping: false ■ Enable Simple Partner Mapping: false ■ Enable Partner NameID Mapping: false Empty fields: User Token Attribute, User Datastore Attribute and Attribute Based User Mapping Also: ■ Partner Token Attribute: NameID ■ Partner Datastore Attribute: username Partner NameID Mapping table will be provisioned with the following entries as NameID format, but without any data in the datastore column: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualified Name ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent User NameID Mapping table will be provisioned with the following entries as NameID format: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualified Name, empty datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName, dn set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress, mail set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent, empty datastore column

Table 4–13 (Cont.) Default Configuration: createWSSValidationTemplate

Token Type	Description
X.509	The Token Mapping section will be created with the following default values: ■ Map token: Map token to Partner ■ Enable Simple User Mapping: false ■ Enable Attribute Based User Mapping: false ■ Enable Simple Partner Mapping: true Empty fields: User Token Attribute, User Datastore Attribute and Attribute Based User Mapping Also: ■ Partner Token Attribute: DN ■ Partner Datastore Attribute: sslclientcertdn
Kerberos	The Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: true ■ Enable Attribute Based User Mapping: false ■ Enable Simple Partner Mapping: false Empty fields: Partner Token Attribute, Partner Datastore Attribute and Attribute Based User Mapping Also: ■ User Token Attribute: TPE_KERBEROS_PRINCIPAL_FULL ■ User Datastore Attribute: mail

4.8.39.2 Syntax

```
createWSSValidationTemplate(templateId, tokenType,
defaultRequesterPPID, description)
```

Argument	Definition
templateId	Specifies the name of the name of the validation template to be created.
tokenType	Specifies the token type of the validation template. Possible values can be: ■ username: indicates that the token type is UsernameToken ■ saml11: indicates that the token type is a SAML 1.1 Assertion ■ saml20: indicates that the token type is a SAML 2.0 Assertion ■ x509: indicates that the token type is an X.509 certificate ■ kerberos: indicates that the token type is a Kerberos token ■ oam: indicates that the token type is OAM
defaultRequesterPPID	Specifies the Requester partner profile to use if OSTS is configured not to map the incoming message to a requester.
description	Specifies an optional description.

4.8.39.3 Example

The following invocation creates a new validation template with default configuration data, and displays the result of the operation.

```
createWSSValidationTemplate(templateId="custom-wss-validtemp", tokenType="custom",
defaultRequesterPPID="requesterPartnerProfileTest", description="custom validation
template")
```

4.8.40 createWSTrustValidationTemplate

Online command that creates a new WS-Trust validation template with default configuration data.

4.8.40.1 Description

Creates a new WS-Trust validation template with default configuration data, and displays the result of the operation.

The WS-Trust validation template is created with the values in Table 4–14, depending on the token type.

Table 4–14 Default Configuration: createWSTrustValidationTemplate

Token Type	Description
Username	The WS-Trust validation template will be created with the following default values: ■ Timestamp Lifespan: 600 seconds ■ Enable Credential Validation: false ■ Validation Source: User Store ■ Token Mapping: Map token to User ■ Enable Simple User Mapping: true ■ USer Datastore Attribute: uid

Table 4–14 (Cont.) Default Configuration: createWSTrustValidationTemplate

Token Type	Description
SAML 1.1 or SAML 2.0	The WS-Trust validation template will be created with the following default values: ■ Authentication Timeout: 3600 seconds ■ Timestamp Lifespan: 3600 seconds The Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: false ■ Enable User NameID Mapping: true ■ Enable Attribute Based User Mapping: false Empty fields: User Datastore Attribute, Attribute Based User Mapping User NameID Mapping table will be provisioned with the following entries as NameID format: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName, empty datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName, dn set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress, mail set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent, empty datastore column
X.509	The WS-Trust Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: true ■ Enable Attribute Based User Mapping: false ■ Enable Simple Partner Mapping: true ■ User Token Attribute: CN ■ User Datastore Attribute: CN ■ Attribute Based User Mapping (empty)
Kerberos	The WS-Trust Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: true ■ Enable Attribute Based User Mapping: false ■ Attribute Based User Mapping (empty) ■ User Token Attribute: TPE_KERBEROS_PRINCIPAL_FULL ■ User Datastore Attribute: mail

Table 4–14 (Cont.) Default Configuration: createWSTrustValidationTemplate

Token Type	Description
OAM	The WS-Trust Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: true ■ Enable Attribute Based User Mapping: false ■ Attribute Based User Mapping (empty) ■ User Token Attribute: TPE_NAME_ID ■ User Datastore Attribute: uid
custom	The WS-Trust Token Mapping section will be created with the following default values: ■ Map token: Map token to None ■ Enable Simple User Mapping: false ■ Enable Attribute Based User Mapping: false ■ Attribute Based User Mapping (empty) ■ User Token Attribute: (empty) ■ User Datastore Attribute: (empty)

4.8.40.2 Syntax

```
createWSTrustValidationTemplate(templateId, tokenType, description)
```

Argument	Definition
templateId	Specifies the name of the name of the WS-Trust validation template to be created.
tokenType	Specifies the token type of the WS-Trust validation template. Possible values can be: ■ username: indicates that the token type is UsernameToken ■ saml11: indicates that the token type is a SAML 1.1 Assertion ■ saml20: indicates that the token type is a SAML 2.0 Assertion ■ x509: indicates that the token type is an X.509 certificate ■ kerberos: indicates that the token type is a Kerberos token ■ oam: indicates that the token type is an Oracle Access Manager token, supported by default ■ <other>: in this case, the token type is assumed to be a custom token type, referenced by <other> (replace <other> by a value)
description	Specifies an optional description.

4.8.40.3 Example

The following invocation creates a new WS-Trust validation template with default configuration data, and displays the result of the operation.

```
createWSTrustValidationTemplate(templateId="custom-wss-validtemp",
tokenType="custom", description="custom validation template")
```

4.8.41 deleteValidationTemplate

Online command that deletes a validation template.

4.8.41.1 Description

Deletes a validation template referenced by the validationTemplateId parameter, and displays the result of the operation.

4.8.41.2 Syntax

```
deleteValidationTemplate(validationTemplateId)
```

Argument	Definition
validationTemplateId	Specifies the name of the validation template to be removed.

4.8.41.3 Example

The following invocation deletes a validation template referenced by the validationTemplateId parameter, and displays the result of the operation.

```
deleteValidationTemplate(validationTemplateId="custom-wss-validtemp")
```

4.9 Oracle Keystore Service

This section contains commands used with the OPSS keystore service.

> **Note:** You need to acquire an OPSS handle to use keystore service commands. For details, see Managing Keys and Certificates with the Keystore Service in the *Oracle Fusion Middleware Security Guide*.

Table 4–15 lists the WLST commands used to manage the keystore service.

Table 4–15 OPSS Keystore Service Commands

Use this Command...	to...
changeKeyPassword	Change the password for a key.
changeKeyStorePassword	Change the password on a keystore.
createKeyStore	Create a keystore.
deleteKeyStore	Delete a keystore.
deleteKeyStoreEntry	Delete an entry in a keystore.
exportKeyStore	Export a keystore to file.
exportKeyStoreCertificate	Export a certificate to a file.
exportKeyStoreCertificateRequest	Export a certificate request to a file.
generateKeyPair	Generate a keypair.
generateSecretKey	Generate a secret key.
getKeyStoreCertificates	Get information about a certificate or trusted certificate.
getKeyStoreSecretKeyProperties	Get the secret key properties.
importKeyStore	Import a keystore from file.
importKeyStoreCertificate	Import a certificate or other object.
listExpiringCertificates	List certificates expiring in a specified period.

Table 4–15 (Cont.) OPSS Keystore Service Commands

Use this Command...	to...
listKeyStoreAliases	List aliases in a keystore.
listKeyStores	List all the keystores in a stripe.

4.9.1 changeKeyPassword

Changes a key password.

4.9.1.1 Description

Changes the password for a key.

4.9.1.2 Syntax

```
svc.changeKeyPassword(appStripe='stripe', name='keystore', password='password',
alias='alias', currentkeypassword='currentkeypassword',
newkeypassword='newkeypassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe containing the keystore
name	Specifies the name of the keystore
password	Specifies the keystore password
alias	Specifies the alias of the key entry whose password is changed
currentkeypassword	Specifies the current key password
newkeypassword	Specifies the new key password

4.9.1.3 Example

This example changes the password on the key entry `orakey`:

```
svc.changeKeyPassword(appStripe='system', name='keystore', password='password',
alias='orakey', currentkeypassword='currentkeypassword',
newkeypassword='newkeypassword')
```

4.9.2 changeKeyStorePassword

Changes the password of a keystore.

4.9.2.1 Description

Changes the password of the specified keystore.

4.9.2.2 Syntax

```
svc.changeKeyStorePassword(appStripe='stripe', name='keystore',
currentpassword='currentpassword', newpassword='newpassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe containing the keystore
name	Specifies the name of the keystore
currentpassword	Specifies the current keystore password
newpassword	Specifies the new keystore password

4.9.2.3 Example

This example changes the password for `keystore2`.

```
svc.changeKeyStorePassword(appStripe='system', name='keystore2',
currentpassword='currentpassword', newpassword='newpassword')
```

4.9.3 createKeyStore

This keystore service command creates a new keystore.

4.9.3.1 Description

Creates a new keystore on the given application stripe.

4.9.3.2 Syntax

```
svc.createKeyStore(appStripe='stripe', name='keystore',
password='password',permission=true|false)
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore is created.
name	Specifies the name of the new keystore.
password	Specifies the keystore password.
permission	This parameter is true if the keystore is protected by permission only, false if protected by both permission and password.

4.9.3.3 Example

This example creates a keystore named `keystore1`.

```
svc.createKeyStore(appStripe='system', name='keystore1', password='password',
permission=true)
```

4.9.4 deleteKeyStore

Deletes the named keystore.

4.9.4.1 Description

This keystore service command deletes a specified keystore.

4.9.4.2 Syntax

```
svc.deleteKeyStore(appStripe='stripe', name='keystore', password='password')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore to be deleted.
password	Specifies the keystore password.

4.9.4.3 Example

This example deletes the keystore named `keystore1`.

```
svc.deleteKeyStore(appStripe='system', name='keystore1', password='password')
```

4.9.5 deleteKeyStoreEntry

Deletes a keystore entry.

4.9.5.1 Description

This command deletes the specified entry in a keystore.

4.9.5.2 Syntax

```
svc.deleteKeyStoreEntry(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the entry to be deleted
keypassword	Specifies the key password of the entry to be deleted

4.9.5.3 Example

This example deletes a keystore entry denoted by alias `orakey`.

```
svc.deleteKeyStoreEntry(appStripe='system', name='keystore2', password='password',
alias='orakey', keypassword='keypassword')
```

4.9.6 exportKeyStore

Exports a keystore to a file.

4.9.6.1 Description

Exports a keystore to the specified file.

4.9.6.2 Syntax

```
svc.exportKeyStore(appStripe='stripe', name='keystore', password='password',

aliases='comma-separated-aliases', keypasswords='comma-separated-keypasswords',
type='keystore-type', filepath='absolute_file_path')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
aliases	Comma separated list of aliases to be exported.
keypasswords	Comma separated list of the key passwords correspo nding to aliases.
type	Exported keystore type. Valid values are 'JKS' or 'JCEKS'.
filepath	Absolute path of the file where keystore is exported.

4.9.6.3 Example

This example exports two aliases from the specified keystore.

```
svc.exportKeyStore(appStripe='system', name='keystore2',
password='password',aliases='orakey,seckey',
keypasswords='keypassword1,keypassword2',
type='JKS',filepath='/tmp/file.jks')
```

4.9.7 exportKeyStoreCertificate

Exports a certificate.

4.9.7.1 Description

Exports a certificate, trusted certificate or certificate chain.

4.9.7.2 Syntax

```
svc.exportKeyStoreCertificate(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword',
type='entrytype',filepath='absolute_file_path')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the entry to be exported
keypassword	Specifies the key password.
type	Specifies the type of keystore entry to be exported. Valid values are 'Certificate', 'TrustedCertificate' or 'CertificateChain'.

Argument	Definition
filepath	Specifies the absolute path of the file where certificate, trusted certificate or certificate chain is exported.

4.9.7.3 Example

This example exports a certificate corresponding to the `orakey` alias:

```
svc.exportKeyStoreCertificate(appStripe='system', name='keystore2',
password='password', alias='orakey', keypassword='keypassword',
type='Certificate', filepath='/tmp/cert.txt')
```

4.9.8 exportKeyStoreCertificateRequest

Exports a certificate request.

4.9.8.1 Description

Generates and exports a certificate request from a keystore.

4.9.8.2 Syntax

```
svc.exportKeyStoreCertificateRequest(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword',
filepath='absolute_file_path')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the entry's alias name.
keypassword	Specifies the key password.
filepath	Specifies the absolute path of the file where certificate request is exported.

4.9.8.3 Example

This example exports a certificate request corresponding to the `orakey` alias.

```
svc.exportKeyStoreCertificateRequest(appStripe='system', name='keystore2',
password='password', alias='orakey', keypassword='keypassword',
filepath='/tmp/certreq.txt')
```

4.9.9 generateKeyPair

Generates a key pair in a keystore.

4.9.9.1 Description

Generates a key pair in a keystore and wraps it in a demo CA-signed certificate.

4.9.9.2 Syntax

```
svc.generateKeyPair(appStripe='stripe', name='keystore', password='password',
```

```
dn='distinguishedname', keysize='keysize', alias='alias',
keypassword='keypassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
dn	Specifies the distinguished name of the certificate wrapping the key pair.
keysize	Specifies the key size.
alias	Specifies the alias of the key pair entry.
keypassword	Specifies the key password.

4.9.9.3 Example

This example generates a keypair in `keystore2`.

```
svc.generateKeyPair(appStripe='system', name='keystore2', password='password',
dn='cn=www.oracle.com', keysize='1024', alias='orakey', keypassword='keypassword')
```

4.9.10 generateSecretKey

Generates a secret key.

4.9.10.1 Description

Generates a symmetric key in a keystore.

4.9.10.2 Syntax

```
svc.generateSecretKey(appStripe='stripe', name='keystore', password='password',
algorithm='algorithm', keysize='keysize', alias='alias',
keypassword='keypassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
algorithm	Specifies the symmetric key algorithm.
keysize	Specifies the key size.
alias	Specifies the alias of the key entry.
keypassword	Specifies the key password.

4.9.10.3 Example

This example generates a keypair with keysize 128 in `keystore2`.

```
svc.generateSecretKey(appStripe='system', name='keystore2', password='password',
```

```
algorithm='AES', keysize='128', alias='seckey', keypassword='keypassword')
```

4.9.11 getKeyStoreCertificates

Gets a certificate from the keystore.

4.9.11.1 Description

Retrieves information about a certificate or trusted certificate.

4.9.11.2 Syntax

```
svc.getKeyStoreCertificates(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the certificate, trusted certificate or certificate chain to be displayed.
keypassword	Specifies the key password.

4.9.11.3 Example

This example gets certificates associated with `keystore3`.

```
svc.getKeyStoreCertificates(appStripe='system', name='keystore3',
password='password', alias='orakey', keypassword='keypassword')
```

4.9.12 getKeyStoreSecretKeyProperties

Retrieves secret key properties.

4.9.12.1 Description

Retrieves secret key properties like the algorithm.

4.9.12.2 Syntax

```
svc.getKeyStoreSecretKeyProperties(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the secret key whose properties are displayed.

Argument	Definition
keypassword	Specifies the secret key password.

4.9.12.3 Example

This example gets properties for secret key `seckey`:

```
svc.getKeyStoreSecretKeyProperties(appStripe='system', name='keystore3',
password='password', alias='seckey', keypassword='keypassword')
```

4.9.13 importKeyStore

Imports a keystore from file.

4.9.13.1 Description

Imports a keystore from a system file.

4.9.13.2 Syntax

```
svc.importKeyStore(appStripe='stripe', name='keystore', password='password',
aliases='comma-separated-aliases', keypasswords='comma-separated-keypasswords',
type='keystore-type', permission=true|false, filepath='absolute_file_path')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
aliases	Specifies the comma-separated aliases of the entries to be imported from file.
keypasswords	Specifies the comma-separated passwords of the keys in file.
type	Specifies the imported keystore type. Valid values are 'JKS' or 'JCEKS'.
filepath	Specifies the absolute path of the keystore file to be imported.
permission	Specifies true if keystore is protected by permission only, false if protected by both permission and password.

4.9.13.3 Example

This example imports a file to `keystore2`:

```
svc.importKeyStore(appStripe='system', name='keystore2',
password='password',aliases='orakey,seckey', keypasswords='keypassword1,
keypassword2', type='JKS', permission=true, filepath='/tmp/file.jks')
```

4.9.14 importKeyStoreCertificate

Imports a certificate or other specified object.

4.9.14.1 Description

Imports a certificate, trusted certificate or certificate chain.

4.9.14.2 Syntax

```
svc.importKeyStoreCertificate(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword',
type='entrytype',filepath='absolute_file_path')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the entry to be imported.
keypassword	Specifies the key password of the newly imported entry.
type	Specifies the type of keystore entry to be imported. Valid values are 'Certificate', 'TrustedCertificate' or 'CertificateChain'.
filepath	Specifies the absolute path of the file from where certificate, trusted certificate or certificate chain is imported.

4.9.14.3 Example

This example imports a certificate into `keystore2`.

```
svc.importKeyStoreCertificate(appStripe='system', name='keystore2',
password='password', alias='orakey', keypassword='keypassword',
type='Certificate', filepath='/tmp/cert.txt')
```

4.9.15 listExpiringCertificates

Lists expiring certificates.

4.9.15.1 Description

Lists expiring certificates and optionally renews them.

4.9.15.2 Syntax

```
svc.listExpiringCertificates(days='days', autorenew=true|false)
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
days	Specifies that the list should only include certificates within this many days from expiration.
autorenew	Specifies true for automatically renewing expiring certificates, false for only listing them.

4.9.15.3 Example

This example lists certificates expiring within one year, and requests that they be renewed:

```
svc.listExpiringCertificates(days='365', autorenew=true)
```

4.9.16 listKeyStoreAliases

Lists the aliases in a keystore.

4.9.16.1 Description

Lists the aliases in a keystore for a given type of entry.

4.9.16.2 Syntax

The syntax is as follows:

```
svc.listKeyStoreAliases(appStripe='stripe', name='keystore',
password='password', type='entrytype')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
type	Specifies the type of entry for which aliases are listed. Valid values are 'Certificate', 'TrustedCertificate', 'SecretKey' or '*'.

4.9.16.3 Example

This example lists secret keys in `keystore2`:

```
svc.listKeyStoreAliases(appStripe='system', name='keystore2',
password='password', type='SecretKey')
```

4.9.17 listKeyStores

Lists all the keystores in a stripe.

4.9.17.1 Description

Lists all the keystores in the specified stripe.

4.9.17.2 Syntax

```
svc.listKeyStores(appStripe='stripe')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe whose keystores are listed.

4.9.17.3 Example

This example lists all keystores on all stripes.

```
svc.listKeyStores(appStripe='*')
```

5

User Messaging Service (UMS) Custom WLST Commands

Use the User Messaging Service commands, listed in Table 5–1, 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 5–1 User Messaging Service for WLST Configuration

Command category	Description
Section 5.1, "UMS WLST Command Group"	Manage Oracle Unified Messaging Service commands.

5.1 UMS WLST Command Group

The UMS WLST commands are listed under the command group "ums".

5.1.1 manageUserMessagingPrefs

Command Category: UMS

Use with WLST: Offline

5.1.1.1 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.

5.1.1.2 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.

5.1.1.3 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')
```

5.1.2 deployUserMessagingDriver

Command Category: UMS

Use with WLST: Online

5.1.2.1 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.

5.1.2.2 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.

5.1.2.3 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')
```

6

DMS Custom WLST Commands

Use the Dynamic Monitoring Service (DMS) commands in the categories in Table 6–1 to view performance metrics and to configure Event Tracing.

> **Note:** To use these DMS custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Table 6–1 DMS Command Categories

Command category	Description
DMS Metric Commands	View information about performance metrics.
DMS Event Tracing Commands	Configure Event Tracing

6.1 DMS Metric Commands

Use the commands in Table 6–2 to view information about 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* and the appendix "Instrumenting Applications with DMS" in the *Oracle Fusion Middleware Performance Guide.*

Table 6–2 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

6.1.1 displayMetricTableNames

Command Category: DMS Metrics

Use with WLST: Online

6.1.1.1 Description

Displays the names of the available DMS metric tables. The returned value is a list of metric table names.

6.1.1.2 Syntax

```
displayMetricTableNames([servers])
```

Argument	Definition

servers	Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic Server instance names and system 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 for all WebLogic servers and system components.

6.1.1.3 Examples

The following example displays metric table names for all WebLogic servers and system 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 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
.
.
.
```

6.1.2 displayMetricTables

Command Category: DMS Metrics

Use with WLST: Online

6.1.2.1 Description

Displays the content of the DMS metric tables.

The returned value is list of DMS metric tables, with the following information about each table:

- The metric table name.
- The metric table schema information.
- The metric table Rows.

The metric table schema information contains the following:

- The name of the column.
- The type of the column value.
- The unit of the column.
- The description of the column.

6.1.2.2 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 metric table name. You can specify multiple metric table names in a comma-separated list. These are the same names output by the WLST command displayMetricTableNames.

Argument	Definition

servers	Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic Server instance names and system 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 for all WebLogic servers and system components.

variables	Optional. Defines the metric aggregation parameters. Valid values are a set of name-value pairs. 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.

6.1.2.3 Examples

The following example displays the data from the JVM and the weblogic.management.runtime.WebAppComponentRuntimeMBean metric tables, and limits it to data retrieved from soa_server1 and bam_server1:

```
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
.
.
.
```

6.1.3 dumpMetrics

Command Category: DMS Metrics

Use with WLST: Online

6.1.3.1 Description

Displays available metrics in the internal format or in XML. The returned value is a text document.

6.1.3.2 Syntax

```
dumpMetrics([servers] [, format])
```

Argument	Definition

servers	Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic Server instance names and system 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 for all WebLogic servers and system components.

format	Optional. Specifies the command output format. Valid values are 'raw' (the default), 'xml, and 'pdml'. For example:
	`format='raw'`
	`format='xml'`
	`format='pdml'`
	DMS raw format is a simple metric display format; it displays one metric per line.

6.1.3.3 Examples

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 soa_server1 and bam_server1 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>
.
.
.
```

6.1.4 reloadMetricRules

Command Category: DMS Metrics

Use with WLST: Online

6.1.4.1 Description

Reloads the metric rules. You must run this command after you deploy system components or after you modify metric rules. Generally, Oracle does not recommend that you modify metric rules.

6.1.4.2 Syntax

```
reloadMetricRules()
```

6.1.4.3 Example

The following example reloads metric rules for all servers running in the domain:

```
reloadMetricRules()
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
.
.
.
```

6.2 DMS Event Tracing Commands

Use the commands in Table 6–3 to configure Event Tracing. Event Tracing configures live tracing with no restarts. DMS metrics that were updated using Oracle Fusion Middleware products may be traced using the DMS Event Tracing feature.

For information about using DMS Event Tracing, see "DMS Tracing and Events" in the *Oracle Fusion Middleware Performance Guide*.

Table 6–3 DMS Tracing Commands

Use this command...	To...	Use with WLST...
addDMSEventDestination	Add a new destination to the Event Tracing configuration.	Online
addDMSEventFilter	Add a filter to the Event Tracing configuration.	Online
addDMSEventRoute	Adds the specified event route to the Event Tracing configuration	Online
enableDMSEventTrace	Enable an event trace and create a filter with a specified condition and destination and an enabled event-route.	Online
listDMSEventConfiguration	Display an overview of the event tracing configuration.	Online

Table 6–3 (Cont.) DMS Tracing Commands

Use this command...	To...	Use with WLST...
listDMSEventDestination	Display the full configuration for a destination or a list of all destinations.	Online
listDMSEventFilter	Displays the configuration of a filter or a list of all filters.	Online
listDMSEventRoutes	Displays event routes and their status (enabled or disabled).	Online
removeDMSEventDestination	Removes the specified destination.	Online
removeDMSEventFilter	Removes the specified filter.	Online
removeDMSEventRoute	Removes the specified event route.	Online
updateDMSEventDestination	Updates configuration of an event destination.	Online
updateDMSEventFilter	Updates the configuration of an event filter.	Online
updateDMSEventRoute	Updates the configuration of an event route.	Online

6.2.1 addDMSEventDestination

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.1.1 Description

Adds a new destination to the Event Tracing configuration. If a destination with the same ID already exists, the command reports this and does not add the destination. You must be connected to the Administration Server to add a destination. If you are not, an error is returned.

6.2.1.2 Syntax

```
addDMSEventDestination(id=id [, name=dest_name] ,class=class_name
              [, props= {'name': 'value'...}] [,server=server_name])
```

Argument	Definition
`id`	The unique identifier for the specified destination.
`name`	Optional. A name for the destination.
`class`	The full class name of the destination. See Table 6–4 for a list of available classes.
`props`	Optional. The name/value properties to use for the destination. Some destinations require properties. For example, the LoggerDestination class requires the property loggerName. See addDMSEventFilter for information about the syntax and allowed values.
`server`	Optional. The server on which to perform this operation. The default is the server to which you are connected.

Table 6–4 shows the built-in destinations, with the full runtime class name.

Table 6–4 Built-In Destinations

Runtime Destination Class Name	Description
oracle.dms.trace2.runtime.LoggerDestination	Uses ODL to send the log messages to a file.
oracle.dms.event.HTTPRequestTrackerDestination	Dumps the set of active HTTP requests, allowing an administrator to get a snapshot of activity.
oracle.dms.jrockit.jfr.JFRDestination	Passes events to the JRockit Flight Recorder so that they can be viewed in the context of other data coming from the JRockit JVM and WLDF using JRockit Mission Control.
oracle.dms.jmx.MetricMBeanFactory	Exposes Nouns as MBeans.
oracle.dms.util.StackTraceCollatorDestination	Collates the stack traces that are in play whenever the events of interest occur. This is primarily a debugging tool. The collated data is written out on shutdown, and also when an event being handled has not been reported for a certain period of time (defaults to one minute).

6.2.1.3 Examples

The following example adds a destination with the ID jfr, the name Flight-Recorder, and the class oracle.dms.event.JRockitFlightRecorder:

```
addDMSEventDestination(id='jfr', name='Flight-Recorder',
                           class='oracle.dms.event.JRockitFlightRecorder')

Destination "jfr" added.
```

The following example adds a destination with the ID destination1, the name File-system, the class oracle.dms.trace2.runtime.LoggerDestination. Because the LoggerDestination requires the property loggerName, it sets the value to trace2-logger:

```
addDMSEventDestination(id='destination1', name='File-system',
                       class='oracle.dms.trace2.runtime.LoggerDestination',
                       props={'loggerName': 'trace2-logger'})

Destination "destination1" added.
```

The following example attempts to add a destination with an ID that already exists:

```
addDMSEventDestination(id='destination1', name='File-system',
                       class='oracle.dms.trace2.runtime.LoggerDestination',
                       props={'loggerName': 'trace2-logger'})

Destination "destination1" already exists. Unable to add this.
```

6.2.2 addDMSEventFilter

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.2.1 Description

Adds a filter to the Event Tracing configuration. If a filter with the same ID already exists, the command returns an error and does not add the filter.

You must be connected to the Administration Server to add an event filter. If you are not, an error message is reported.

6.2.2.2 Syntax

```
addDMSEventFilter(id=id [, name=name] [, etypes]
                  , props= {'prop-name': 'value'...}
                  [, server=server_name])
```

Argument	Definition
id	The unique identifier for specified filter.
name	Optional. The name of the filter.
etypes	Optional. A string containing a comma-separated list of event/action pairs. This argument allows you to create a filter with a broader granularity when used with a condition. It also allows you to create a filter with a broader range of metrics. For example, all nouns or all nouns with the action create.
props	*prop-name*: The name of the filter property. <condition> is the only valid property, and only one condition may be specified. value: The value of the property of the filter.
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

The following shows the syntax for etypes:

```
<etypes>:==
<type>:[<action>]
```

The following lists the valid etypes:

```
EXECUTION_CONTEXT
EXECUTION_CONTEXT:START
EXECUTION_CONTEXT:STOP
HTTP_REQUEST
HTTP_REQUEST:START
HTTP_REQUEST:STOP
NOUN
NOUN:CREATE
NOUN:DELETE
STATE_SENSOR
STATE_SENSOR:CREATE
STATE_SENSOR:DELETE
```

The following shows an etype with two event/action pairs, separated by a comma:

```
etypes='NOUN:DELETE, STATE_SENSOR:DELETE'
```

The following shows the syntax for the <condition> property of the argument props. The arguments are described in the tables following the syntax:

```
<condition>::=
<type>  [<operator> <condition>]

<type>::=
<nountype> | <context>
```

```
<nountype>::=
NOUNTYPE <nountype-operator> value

<nountype-operator>::=
"equals" | "starts_with" | "contains" | "not_equals"

<context>::=
CONTEXT <name> <context-operator> [<value>] [IGNORECASE=true|false]
[DATATYPE="string|long|double"
]
<context-operator>::=
"equals" | "starts_with" | "contains" | "not_equals" | "is_null" | "gt" | "le" |
"ge"

<operator>::=
 AND |OR
```

The following table describes the arguments for <type>:

Value	Description
<nountype>	Each Sensor, with its associated metric, is organized in a hierarchy according to Nouns. A Noun type is a name that reflects the set of metrics being collected. For example, JDBC could be a Noun type. For information about Sensors and Nouns, see "Understanding DMS Terminology (Nouns and Sensors)" in the *Oracle Fusion Middleware Performance Guide*
<context>	An Execution Context is an association of the Execution Context ID (ECID), Relationship ID (RID), and Maps of Values. This argument allows the data stored in the map of values to be inspected and used by the filter. For example, if the map contains the key "user", you can create a filter that returns requests with "user" equal to "bruce".

The following table describes the arguments for <nountype>:

Value	Description
NOUNTYPE	A keyword.
<nountype-operator>	The following are valid operators: ■ equals: Filters only if the Noun type name equals the value. ■ starts_with: Filters only if the Noun type name starts with the value. ■ contains: Filters only if the Noun type name equals the value. ■ not_equals: Filters only if the Noun type name does not equal the value.
value	The name of the Noun type on which to operate. The name can be any object for which you want to measure performance.

The following table describes <context>

Value	Description
CONTEXT	A keyword.
name	The name of the context to filter.

Value	Description
value	The name of the context on which to operate.
<context-operator>	The following are valid operators: ■ equals: Filters only if the context name equals the value. ■ starts_with: Filters only if the context name starts with the value. ■ contains: Filters only if the context name equals the value. ■ not_equals: Filters only if the context name does not equal the value. ■ is_null: Filters only if the context name is null. ■ lt: Filters only if the context name is less than the value. ■ gt: Filters only if the context name is greater than the value. ■ le: Filters only if the context name is less than or equal to the value. ■ ge: Filters only if the context name is greater than or equal to the value.
IGNORECASE	Optional. If specified, the case of the value is ignored.
DATATYPE	Optional. The valid values are string, long, or double. The default is string.

6.2.2.3 Examples

The following example adds a filter with the name MyFilter, specifying a Noun type and context:

```
addDMSEventFilter(id='mds1', name='MyFilter',
        props={'condition': 'NOUNTYPE equals MDS_Connections AND CONTEXT user
equals bruce IGNORECASE'})

Filter "mds1" added.
```

The following example attempts to add a filter with the same id. The command returns an error:

```
addDMSEventFilter(id='mds1', name='MyFilter',
      props={'condition': 'NOUNTYPE equals MDS_Connections AND CONTEXT user equals
bruce'})

Filter "mds1" already exists. Unable to add this.
```

The following example adds a filter with two event/action pairs:

```
addDMSEventFilter(id='mds2', name='MyFilter',
          etypes='NOUN:CREATE,HTTP_REQUEST:START',
          props={'condition': 'NOUNTYPE equals MDS_Connections
                  AND CONTEXT user equals bruce IGNORECASE=true'})
 Filter "mds2" added.
```

6.2.3 addDMSEventRoute

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.3.1 Description

Adds the specified event route to the Event Tracing configuration. If an event route with the same ID already exists, the command returns an error and does not add the event route.

You must be connected to the Administration Server to add an event route. If you are not, an error is returned.

6.2.3.2 Syntax

```
addDMSEventRoute([filterid=filter_id], destinationid=destination_id,
[enable=true|false] [,server=server_name])
```

Argument	Definition
`filterid`	Optional. The unique identifier for the filter.
`destinationid`	The unique identifier for the specific destination. The destination must exist.
`enable`	Optional. Enables the filter. Valid values are `true` and `false`. The default is true.
`server`	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.3.3 Examples

The following example adds an event route with the filter id of mds1 and the destination id of jrf:

```
addDMSEventRoute(filterid='mds1', destinationid='jfr', enable='false')
Event-route for filter "mds1", destination "jfr" added.
```

The following example attempts to add an event route that already exists:

```
addDMSEventRoute(filterid='mds1', destinationid='jfr', enable='false')
Event-route for filter "mds1", destination "jfr" already exists. Unable to add
this.
```

6.2.4 enableDMSEventTrace

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.4.1 Description

Enables an event trace and creates a filter with a specified condition and destination and an enabled event-route. This is a simple way to start filtering, without having to explicitly create a filter, destination and event-route, but with less configuration options. The specified destination must exist.

You must be connected to the Administration Server to enable a DMS event trace. If you are not, an error is returned.

If you require a more complex configuration, use the addDMSEventDestination, addDMSEventFilter, and addDMSEventRoute.

6.2.4.2 Syntax

```
enableDMSEventTrace(destinationid=destinationid [, etypes=etype]
                    [, condition=condition]  [, server=server_name])
```

Argument	Definition
destinationid	The unique identifier for the specific destination. Any existing destination is valid.
etypes	Optional. A string containing a comma-separated list of event/action pairs. See addDMSEventFilter for a list of available etypes.
condition	Optional. A condition on which to filter. See addDMSEventFilter for the syntax for a condition. If no condition is specified, all DMS events will be passed
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.4.3 Example

The following example enables an event trace with a specified condition:

```
enableDMSEventTrace(condition='CONTEXT username EQUALS Joe AND CONTEXT ip EQUALS
192.168.1.5')

Filter "EventTrace9", using Destination "LoggerDestination" added, and event-route
enabled.
```

6.2.5 listDMSEventConfiguration

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.5.1 Description

Displays an overview of the Event Tracing configuration.

6.2.5.2 Syntax

```
listDMSEventConfiguration([server=server_name]]
```

Argument	Definition
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.5.3 Example

The following example lists the configuration for the Managed Server to which you are connected:

```
listDMSEventConfiguration()

Event routes:
  FILTER      DESTINATION
  MyFilter    des1
  MyFilter    des2
  null        des3

Filters with no event route:
  Fred

Destinations with no event route:
  des4
```

6.2.6 listDMSEventDestination

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.6.1 Description

For a specific destination, display the full configuration. If no destination ID is specified, list the destination ID and name for all the destinations in the Event Tracing configuration.

6.2.6.2 Syntax

```
listDMSEventDestination([id=id] [, server=server_name)
```

Argument	Definition
`id`	Optional. The unique identifier for the specific destination.
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.6.3 Examples

The following example displays information about the destinations for the Managed Server to which you are connected:

```
listDMSEventDestination()
ID  : destination1
NAME: File-system
ID  : jrf
NAME: Flight-Recorder
```

The following example displays information about the destinations for the Managed Server, MS1:

```
listDMSEventDestination(server='MS1')
ID        NAME
Network1  Send file over network
desman1   File-system
```

The following example displays information about the destination destination1:

```
listDMSEventDestination(id='destination1')
ID: destination1
NAME: File-system
CLASS: oracle.dms.trace2.runtime.LoggerDestination
PROPERTIES:
NAME       VALUE
LoggerName trace2-logger
```

6.2.7 listDMSEventFilter

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.7.1 Description

For a specific filter, displays the full configuration. If you do not specify a filter ID, the command displays the filter ID and name for all the filters in the Event Tracing configuration.

6.2.7.2 Syntax

```
listDMSEventFilter([id=id] [, server=server_name])
```

Argument	Definition
id	Optional. The unique identifier for specified filter.
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.7.3 Example

The following example displays the list of all the filters in the Event Tracing configuration:

```
listDMSEventFilter()

ID    NAME
mds1  MyFilter
mds2  MDS2Filter
```

The following example displays the configuration of the filter mds1:

```
listDMSEventFilter(id='mds1')

ID  : mds1
NAME: MyFilter
PROPERTIES
CONDITION: NOUNTYPE equals MDS_Connections AND CONTEXT user equals
bruce IGNORECASE=false
```

6.2.8 listDMSEventRoutes

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.8.1 Description

List the events routes and their status (enabled or disabled) that are associated with the specified filter or destination. If you do not specify a filterid or destinationid, this command lists all the event routes in the Event Tracing configuration.

6.2.8.2 Syntax

```
listDMSEventRoutes([filterid=filter_id] [, destinationid=destination_id]
                   [, server=server_name])
```

Argument	Definition
filterid	Optional. The unique identifier for the filter.
destinationid	Optional. The unique identifier for the specific destination. The destination must exist.

Argument	Definition
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.8.3 Examples

The following example lists all event routes:

```
listDMSEventRoutes()
  FILTER     : mdsbruce
  DESTINATION: jfr
  ENABLED    : false
  FILTER     : null
  DESTINATION: destination1
  ENABLED    : true
```

The following example lists the event routes with the filter id of filter1:

```
listDMSEventRoutes(filterid='filter1')
  FILTER     : filter1
  DESTINATION: jfr
  ENABLED    : true
  FILTER     : filter1
  DESTINATION: destination1
  ENABLED    : true
```

The following example lists the event routes with the destination id of destination1:

```
listDMSEventRoutes(destinationid='destination1')
  FILTER     : filter1
  DESTINATION: destination1
  ENABLED    : true
```

6.2.9 removeDMSEventDestination

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.9.1 Description

Removes an existing destination from the Event Tracing configuration. You can remove a destination only if no event route depends on the destination. If an event route that depends on the destination exists, a warning is returned.

You must be connected to the Administration Server to remove a destination. If you are not, an error is returned.

6.2.9.2 Syntax

```
removeDMSEventDestination(id=id [, server=server_name])
```

Argument	Definition
id	The unique identifier for the destination to be removed.
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.9.3 Examples

The following example removes the destination jfr:

```
removeDMSEventDestination(id='jfr')

Destination "jfr" removed.
```

The following example attempts to remove the destination styx.inpass.db1. However, because an event route exists for the destination, the command returns an error.

```
removeDMSEventDestination(id='styx.inpass.db1')

Destination "'styx.inpass.db1'" cannot be removed. An event-route currently
exists for that destination. Remove the event-route first using the command
removeDMSEventRoute().
```

6.2.10 removeDMSEventFilter

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.10.1 Description

Removes an existing filter from the Event Tracing configuration. You can remove a filter only if no event route depends on the filter. If an event route that depends on the filter exists, a warning is returned.

You must be connected to the Administration Server to remove an event filter. If you are not, an error is returned.

6.2.10.2 Syntax

removeDMSEventFilter(id=*id* [, server=*server_name*])

Argument	Definition
id	The unique identifier for the filter to be removed.
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.10.3 Example

The following example removes the filter mds1:

```
removeDMSEventFilter(id='mds1')

Filter "mds1" removed.
```

The following example attempts to remove a filter for which and event-route currently exists:

```
removeDMSEventFilter(id='allaccounts')

Filter "allaccounts" cannot be removed. An event-route currently exists for that
filter. Remove the event-route first using the command removeDMSEventRoute().
```

6.2.11 removeDMSEventRoute

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.11.1 Description

Removes the specified event route. You must be connected to the Administration Server to add an event route. If you are not, an error is returned.

6.2.11.2 Syntax

```
removeDMSEventRoute([filterid=filter_id] [, destinationid=destination_id]
                    [, server=server_name])
```

Argument	Definition
`filterid`	Optional. The unique identifier for the filter.
`destinationid`	Optional. The unique identifier for the specific destination. The destination must exist.
`server`	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.11.3 Example

The following example removes the event route with the filterid mds1 and the destination jfr:

```
removeDMSEventRoute(filterid='mds1', destinationid='jfr')
Event-route for filter "mds1", destination "jfr" removed
```

The following example removes the event route with the destination destination1:

```
removeDMSEventRoute(destinationid='destination1')
Event-route for filter "None", destination "destination1" removed
```

6.2.12 updateDMSEventDestination

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.12.1 Description

Updates an existing destination, allowing a specified argument to be updated. You must be connected to the Administration Server to update a destination. If you are not, an error is returned.

6.2.12.2 Syntax

```
updateDMSEventDestination(id=id [, name=dest_name], class=class_name
                [,props= {'name': 'value'...}] [, server=server_name)
```

Argument	Definition
`id`	The unique identifier for the destination to be updated.
`name`	Optional. A name for the destination.
`class`	The full classname of the destination. See Table 6–4 for a list of available destinations.

Argument	Definition
`props`	Optional. The name/value properties to use for the destination. You can add a new property, or update or remove an existing one. If you update properties, you must specify all properties. If you omit a property, it is removed. For example, if a destination contains the properties LoggerName and severity, and you omit severity, it will be removed from the destination. See addDMSEventFilter for information about the syntax and allowed values.
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.12.3 Examples

The following example updates the name of the destination jfr:

```
updateDMSEventDestination(id='jfr', name='Alternative Flight-Recorder')

Destination "jfr" updated.
```

The following example attempts to update a destination that does not exist. The command returns an error:

```
updateDMSEventDestination(id='destination1',
          props={'loggerName': 'MyNewTrace2-logger'})

Destination "destination1" does not yet exist. Unable to update this.
```

6.2.13 updateDMSEventFilter

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.13.1 Description

Updates an existing filter in the Event Tracing configuration.

You must be connected to the Administration Server to update an event filter. If you are not, an error is returned.

6.2.13.2 Syntax

```
updateDMSEventFilter(id=id [, name=name] [,etypes=etypes],
                     props= {'prop-name': 'value'...}
                     [,server=server_name])
```

Argument	Definition
`id`	The unique identifier for the filter to be updated.
`name`	Optional. The name of the filter to be updated.
`etypes`	Optional. A string containing a comma-separated list of event/action pairs. See addDMSEventFilter for a list of valid values.
`props`	*prop-name*: The name of the filter property. `<condition>` is the only valid property, and only one condition may be specified. See addDMSEventFilter for information on the syntax of *prop-name*. *value*: The value of the property of the filter.
server	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.13.3 Examples

The following example updates the filter properties for the filter with the id mds1:

```
updateDMSEventFilter(id='mds1',
 props={'condition': 'NOUNTYPE equals XYZ_Total_Connections AND CONTEXT user
equals bruce'})

Filter "mds1" updated.
```

The following example attempts to update a filter that does not exist:

```
updateDMSEventFilter(id='Filter2')

Filter "Filter2" does not yet exist. Unable to update this.
```

6.2.14 updateDMSEventRoute

Command Category: DMS Event Tracing

Use with WLST: Online

6.2.14.1 Description

Enables or disables the specified event route. You must be connected to the Administration Server to update an event route. If you are not, an error is returned.

6.2.14.2 Syntax

```
updateDMSEventRoute([filterid=filter_id], destinationid=destination_id
        [, enable=true|false] [, server=server_name])
```

Argument	Definition
`filterid`	Optional. The unique identifier for the filter.
`destinationid`	Optional. The unique identifier for the specific destination. The destination must exist.
`enable`	Optional. Enables the filter. Valid values are `true` and `false`.
`server`	Optional. The server on which to perform this operation. The default is the server to which you are connected.

6.2.14.3 Example

The following example disables the event route with the filterid mds1 and the destinationid jfr:

```
updateDMSEventRoute(filterid='mds1', destinationid='jfr', enable='false')
Event-route for filter "mds1", destination "jfr" disabled .
```

7

Logging Custom WLST Commands

Use the logging commands to configure settings for log files and to view and search log files. Table 7–1 describes the different categories of logging commands.

For additional details about configuring and searching log files, see "Managing Log Files and Diagnostic Data" in the *Oracle Fusion Middleware Administrator's Guide*.

> **Note:** To use these logging custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Table 7–1 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.
Selective Tracing Commands	Configure and use selective tracing, which specifies that messages are traced for specific server, loggers, or users.

7.1 Log Configuration Commands

Use the commands in Table 7–2 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 7–2 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

7.1.1 configureLogHandler

Command Category: Log Configuration

Use with WLST: Online

7.1.1.1 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.

7.1.1.2 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 instance, or a string describing a system component. For system 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 maximum file size 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. Note that this options does not apply to the QuickTrace handler. ■ **maxLogSize**—The value of the maximum size of the log files 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). Note that this options does not apply to the QuickTrace handler. ■ **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. Note that this options does not apply to the QuickTrace handler. ■ **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. Note that this options does not apply to the QuickTrace handler.

Argument	Definition
options (continued)	■ **retentionPeriod**—The amount of time 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. Note that this options does not apply to the QuickTrace handler. ■ **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. Note that this options does not apply to the QuickTrace handler. ■ **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. ■ **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. ■ **level**—A Java or ODL level value. The handler level will be set to the given level. ■ **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. ■ **useParentHandlers**—A boolean value. Sets the useParentHandlers flag on the loggers defined by the addToLogger or removeFromLogger options.

The following table lists the properties for the quicktrace-handler. This handler allows you to trace messages from specific loggers and store the messages in memory. For more information, see "Configuring QuickTrace" in the *Oracle Fusion Middleware Administrator's Guide.*

QuickTrace Property	Description
bufferSize	The approximate size of the circular QuickTrace buffer, in which log records are stored in memory. Note that actual memory consumption may be less than, but not more than this value.

QuickTrace Property	Description
enableDMSMetrics	If specified as true, DMS metrics are enabled for the quicktrace-handler. The default is true.
enableUserBuffer	If specified as true, the handler maintains an individual buffer for each user specified in the reserveBufferUserID property. If the user is not defined in the reserveBufferUserID property, the messages are cached in the COMMON buffer. If specified as false, the handler maintains only one buffer, COMMON. The default is false.
flushOnDump	If specified as true, the buffer is flushed when you execute the executeDump command. The default is true.
includeMessageArguments	If specified as true, message arguments are included with the formatted log messages that have a message ID. The default is false.
maxFieldLength	The maximum length, in bytes, for each field in a message. The fields can include the message text, supplemental attributes, thread name, source class name, source method name, and message arguments. The default is 240 bytes. A small number can restrict the amount of information returned for a message. An excessively number can reduce the amount of log records in the buffer because each message uses more bytes.
reserveBufferUserID	A list of user IDs, separated by a comma. If enableUserBuffer is specified as true, any log messages related to the user are written to a separate buffer.
supplementalAttributes	A list of supplemental attribute names. The attributes are listed in the logging.xml file. Setting supplemental attributes requires additional memory or CPU time.
useDefaultAttributes	If specified as true, default attribute values are added to each log message. The default attributes are HOST_ID, HOST_NWADDR, and USER_ID.
useLoggingContext	If specified as true, the log message includes DMS logging context attributes. The default is false. If you enable this option, the trace requires additional CPU time.
useRealThreadID	If specified as true, the handler attempts to use the real thread ID instead of the thread ID that is provided by the jave.util.logging.logRecord. The default is false. If you enable this option, the trace requires additional CPU time.
useThreadName	If specified as true, the log message includes the thread name instead of the thread ID. The default is false.

7.1.1.3 Examples

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'])
```

The following example configures the quicktrace-handler, adding the logger oracle.adf.faces, and enabling user buffers for user1 and user2:

```
configureLogHandler(name="quicktrace-handler", addToLogger="oracle.adf.faces",
       propertyName="enableUserBuffer", propertyValue="true",
      propertyName="reserveBufferUserID", propertyValue="user1, user2")
```

The oracle.adf logger is associated with the handlers odl-handler, wls-domain, and console-handler. When you set the level of the logger, these handlers will use the same level (TRACE:1) for the logger oracle.adf. As a result, much information will be written to the log files, consuming resources. To avoid consuming resources, set the level of the handlers to a lower level, such as WARNING or INFORMATION. For example:

```
configureLogHandler(name="odl-handler", level="WARNING:1")
configureLogHandler(name="wls-domain", level="WARNING:1")
configureLogHandler(name="console-handler", level="WARNING:1")
```

7.1.2 getLogLevel

Command Category: Log Configuration

Use with WLST: Online

7.1.2.1 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.

7.1.2.2 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 instance, or a string describing a system component. For system 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).

7.1.2.3 Examples

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, not runtime 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')
```

7.1.3 listLoggers

Command Category: Log Configuration

Use with WLST: Online

7.1.3.1 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.

7.1.3.2 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 instance, or a string describing a system component. For system 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 (runtime).

7.1.3.3 Examples

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")
```

7.1.4 listLogHandlers

Command Category: Log Configuration

Use with WLST: Online

7.1.4.1 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.

7.1.4.2 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 instance, or a string describing a system component. For system 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.

7.1.4.3 Examples

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")
```

7.1.5 setLogLevel

Command Category: Log Configuration

Use with WLST: Online

7.1.5.1 Description

Sets the level of information written by a given Java logger to a log file.

7.1.5.2 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 instance, or a string describing a system component. For system 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. This option is deprecated for runtime mode. Adding a runtime logger may have no effect because the logger may be garbage collected. If you need to set the level for a logger that has not yet been created, use the persist mode. ■ **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 a system component that does not support changing runtime loggers, this option is ignored. Note: Because runtime loggers may be garbage collected, you should change the level of the runtime logger only if you know that the logger exists and that there is a strong reference to the logger. If the logger is garbage collected, any changes made to the logger level in runtime mode that are not persisted may be lost. ■ **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.

7.1.5.3 Examples

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)
```

7.2 Search and Display Commands

Use the commands in Table 7–3 to view Oracle Fusion Middleware log files and to search log files for particular messages.

Table 7–3 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

7.2.1 displayLogs

Command Category: Search and Display

Use with WLST: Online or Offline

7.2.1.1 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.

7.2.1.2 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 instance, or a system component. For a system 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**—Defines the path to the ORACLE_INSTANCE or WebLogic domain home. The command is executed in disconnected mode when you use this parameter. ▪ **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. ■ **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.

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 system components.

7.2.1.3 Examples

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 a system component, ohs1:

```
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")
```

7.2.2 listLogs

Command Category: Search and Display

Use with WLST: Online or Offline

7.2.2.1 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.

7.2.2.2 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 instance, or an Oracle Fusion Middleware system component. For a system 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**—Defines the path to the ORACLE_INSTANCE or WebLogic domain home. The command is executed in disconnected mode when you use this parameter. ■ **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.

7.2.2.3 Examples

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")
```

7.3 Selective Tracing Commands

Use the commands in Table 7–4 to configure and use selective tracing. Selective tracing provides fine-grained logging for specified users or other properties of a request. In the Use with WLST column, online means the command can only be used when connected to a running server.

Table 7–4 Tracing Commands

Use this command...	To...	Use with WLST...
configureTracingLoggers	Configures one or more loggers for selective tracing.	Online
listActiveTraces	Lists the active traces.	Online
listTracingLoggers	Lists the loggers that support selective tracing.	Online
startTracing	Starts a selective tracing sessions.	Online
stopTracing	Stops one or more selective tracing sessions.	Online

7.3.1 configureTracingLoggers

Command Category: Tracing

Use with WLST: Online

7.3.1.1 Description

Configures one or more loggers for selective tracing. This command also enables or disables a logger for selective tracing.

7.3.1.2 Syntax

```
configureTracingLoggers([options])
```

Argument	Definition
options	A comma-separated list of options, specified as name-value pairs. Valid options are: ■ **target**—Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled will be configured for tracing. ■ **pattern**—A regular expression pattern that is used to filter logger names. The default value matches all tracing logger names. ■ **action**---Enables or disables all loggers for tracing. Valid values are `enable` and `disable`. This option is required; there is no default value.

7.3.1.3 Examples

The following example configures selective tracing for all loggers beginning with `oracle.security`:

```
configureTracingLoggers(pattern='oracle.security.*', action="enable")
Configured 62 loggers
```

The following example disables selective tracing for all loggers:

```
configureTracingLoggers(action="disable")
Configured 1244 loggers
```

7.3.2 listActiveTraces

Command Category: Tracing

Use with WLST: Online

7.3.2.1 Description

Lists the active traces.

7.3.2.2 Syntax

```
listActiveTraces([options])
```

Argument	Definition
options	A comma-separated list of options, specified as name-value pairs. Valid options are: ■ **target**—Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled are listed.

7.3.2.3 Example

The following example lists the active traces:

```
listActiveTraces()
-------------------------------------+----------+----------+------+----------+----------
Trace ID                             |Attr. Name|Attr. Value| Level| Exp. Time |Description
-------------------------------------+----------+----------+------+----------+----------
a9580e65-13c4-420b-977e-5ba7dd88ca7f |USER_ID   |user1      | FINE |           |
a04b47f7-2830-4d80-92ee-ba160cdacf6b |USER_ID  | user2     | FINE |           |
```

7.3.3 listTracingLoggers

Command Category: Tracing

Use with WLST: Online or Offline

7.3.3.1 Description

Lists the loggers that support selective tracing. This command displays a table of logger names and their tracing status. The status `enabled` means that the logger is enabled for tracing on all servers. The status `disabled` means that the logger is disabled for tracing on all servers. The status `mixed` means that the logger is enabled for tracing on some servers, but disabled on others.

7.3.3.2 Syntax

```
listTracingLoggers([options])
```

Argument	Definition
options	A comma-separated list of options, specified as name-value pairs. Valid options are: ■ **target**—Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled are listed. ■ **pattern**—A regular expression pattern that is used to filter logger names. The default value matches all tracing logger names.

7.3.3.3 Example

The following example lists all tracing loggers beginning with `oracle.security`:

```
listTracingLoggers(pattern="oracle.security.*")
-----------------------------------------------------------------------+--------
Logger                                                                 | Status
```

```
------------------------------------------------------------------+--------
oracle.security                                                   | enabled
oracle.security.audit.logger                                      | enabled
oracle.security.jps.az.common.util.JpsLock                        | enabled
 .
 .
 .
```

7.3.4 startTracing

Command Category: Tracing

Use with WLST: Online

7.3.4.1 Description

Starts a new selective tracing session for a specified user or DMS context attribute at a specified level of tracing.

7.3.4.2 Syntax

```
startTracing([options])
```

Argument	Definition
options	A comma-separated list of options, specified as name-value pairs. Valid options are: ■ **target**—Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled are included in the trace. ■ **traceId**—Optional. An identifier for the tracing session. If a traceId is not provided, the command generates a unique traceId. ■ **attrName**—Optional, unless the user argument is not specified. Valid values are USER _ID, APP, CLIENT_HOST, CLIENT_ADDR, composite_ name, WEBSERVICE.name, WEBSERVICE_PORT.name. ■ **attrValue**—Required if attrName is specified. The value of the attribute. ■ **user**—The user name. Messages associated with the user are returned. This is equivalent to passing the USER_ID with the attrName and AttrValue options. ■ **level**—Required. The tracing level. The level must be a valid Java or ODL level. See the table "Mapping of Log Levels Among ODL, Oracle WebLogic Server, and Java" in the *Oracle Fusion Middleware Administrator's Guide*. ■ **desc**—Optional. A description of the tracing session.

7.3.4.3 Example

The following example starts a trace for messages associated with user1 and sets the level of information to FINE:

```
startTracing(user="user1",level="FINE")
Started tracing with ID: 885649f7-8efd-4a7a-9898-accbfc0bbba3
```

7.3.5 stopTracing

Command Category: Tracing

Use with WLST: Online

7.3.5.1 Description

Stops one or more selective tracing sessions.

7.3.5.2 Syntax

```
stopTracing([options])
```

Argument	Definition
options	A comma-separated list of options, specified as name-value pairs. Valid options are: ■ **target**—Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled are included in the operation. ■ **stopAll**---A Jython boolean value (0 or 1) that determines if all of the active traces are stopped. Required if the traceId, user, or attrName and attrValue arguments are not specified. The default value is 0 (false). ■ **traceId**---An identifier for the tracing session to be stopped. Required if the stopAll, user, or attrName and attrValue arguments are not specified. ■ **attrName**---Valid values are USER _ID, APP, CLIENT_HOST, CLIENT_ADDR, composite_name, WEBSERVICE.name, WEBSERVICE_PORT.name. Required if the traceId, user, stopAll arguments are not specified. ■ **attrValue**---Required if attrName is specified. The value of the attribute. ■ **user**---The user name. All tracing sessions associated with the user are stopped. Required if the stopAll, traceId, or attrName and attrValue arguments are not specified.

7.3.5.3 Examples

The following example stops a tracing session with a specified traceId:

```
stopTracing(traceId="a04b47f7-2830-4d80-92ee-ba160cdacf6b")
Stopped 1 traces
```

The following example stops all tracing sessions:

```
stopTracing(stopAll=1)
Stopped 1 traces
```

8

Metadata Services (MDS) Custom WLST Commands

Use the Oracle Metadata Services (MDS) commands in the categories listed in Table 8–1 to manage 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 MDS custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide*.

Table 8–1 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.
Sandbox Metadata Management Commands	Manage the metadata in a sandbox in the MDS repository.
Application Label Management Commands	Manage the labels for the application.
Application Management Deployment Commands	Manage the application deployment.
Multitenancy Management Commands	Manage tenants.

8.1 Repository Management Commands

Use the MDS commands listed in Table 8–2 to manage the MDS repository. In the Use with WLST column, online means the command can only be used when connected to a running Administration 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 Repository Management Commands

Use this command...	To...	Use with WLST...
createMetadataPartition	Create a metadata repository partition.	Online
deleteMetadataPartition	Delete a metadata repository partition.	Online
deregisterMetadataDBRepository	Deregister a database-based MDS repository.	Online
registerMetadataDBRepository	Register a database-based MDS repository.	Online

8.1.1 createMetadataPartition

Command Category: Repository Management

Use with WLST: Online

8.1.1.1 Description

A metadata repository is used as a common repository for managing metadata of different applications. Many applications use the MDS repository to manage their metadata. Each deployed application uses a logical partition in metadata repository. This logical partition also helps in maintaining the metadata lifecycle. Before deploying a application, you create a partition for it in MDS repository. This command creates a partition with the given name in the specified repository.

8.1.1.2 Syntax

```
createMetadataPartition(repository, partition)
```

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.

8.1.1.3 Example

The following example creates the metadata partition `partition1` in the repository `mds-myrepos`:

```
wls:/weblogic/serverConfig> createMetadataPartition(repository='mds-myrepos',
                              partition='partition1')
Executing operation: createMetadataPartition
Metadata partition created: partition1
"partition1"
```

8.1.2 deleteMetadataPartition

Command Category: Repository Management

Use with WLST: Online

8.1.2.1 Description

Deletes a metadata partition in the specified repository. When you delete a repository partition, all of the metadata in that partition is lost.

8.1.2.2 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.

8.1.2.3 Example

The following example deletes the metadata partition `partition1` from the repository `mds-myrepos`:

```
wls:/weblogic/serverConfig> deleteMetadataPartition(repository='mds-myrepos',
                                partition='partition1')
Executing operation: deleteMetadataPartition
Metadata partition deleted: partition1
```

8.1.3 deregisterMetadataDBRepository

Command Category: Repository Management

Use with WLST: Online

8.1.3.1 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.

8.1.3.2 Syntax

```
deregisterMetadataDBRepository(name)
```

Argument	Definition
name	The name of the repository to deregister.

8.1.3.3 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.
```

8.1.4 registerMetadataDBRepository

Command Category: Repository Management

Use with WLST: Online

8.1.4.1 Description

A database metadata repository should be registered with WebLogic Server instances before the application can use it. This command registers a System JDBC data source with the domain for use as database-based metadata repository.

8.1.4.2 Syntax

```
registerMetadataDBRepository(name, dbVendor, host, port, dbName, user, password
[, targetServers])
```

Argument	Definition
name	The name of the repository to register. If the name you supply does not begin with mds-, the commands adds the prefix mds-.
dbVendor	The database vendor. The acceptable values are ORACLE, MSSQL, IBMDB2, and MYSQL.
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 Server instances 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. Register the repository with all Managed Servers to which the application will be deployed.

8.1.4.3 Example

The following example registers the metadata repository `myrepos` to two servers, and specifies 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'
```

8.2 Application Metadata Management Commands

Use the commands in Table 8–3 to manage application metadata.

Table 8–3 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

8.2.1 deleteMetadata

Command Category: Application Metadata

Use with WLST: Online

8.2.1.1 Description

Deletes the selected documents from the application repository. When this command is run against repositories that support versioning (that is, database-based repositories), 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.

8.2.1.2 Syntax

```
deleteMetadata(application, server, docs [, restrictCustTo] [, excludeAllCust]
 [, excludeBaseDocs] [, excludeExtendedMetadata] [, cancelOnException] [,
applicationVersion] [, tenantName])
```

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	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 asterisk (*) represents all documents under the current namespace. The double asterisk (**) represents all documents under the current namespace and also recursively includes all documents in subnamespaces. 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. Valid values are percent (%) or a list of comma-separated customization layer names used to restrict the delete operation to delete only customization documents that match the specified customization layers. Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas.
	For example:
	`restrictCustTo="user[scott]"`
	`restrictCustTo="site[site1],user[scott]"`
	`restrictCustTo="site[site1, %_2],user[scott, m%]"`
	If you do not specify this argument, only customization classes declared in the cust-config element of adf-config.xml are deleted. If there is no cust-config element declared in adf-config.xml, all customization classes are deleted.
	If you specify percent (%) as the value of this argument, all customizations are deleted, whether or not they are declared in the cust-config element of adf-config.xml.
	Use this option to delete all customizations or a subset of declared customizations. You can also use this option to delete customizations from customization classes that are not declared in the cust-config element of adf-config.xml.

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.

Argument	Definition
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.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.2.1.3 Examples

The following example deletes metadata files under the package `mypackage` from `mdsApp` deployed in the server `server1`:

```
wls:/weblogic/serverConfig> deleteMetadata(application='mdsapp',
          server='server1', docs='/mypackage/*')
Executing operation: deleteMetadata.
"deleteMetadata" operation completed. Summary of "deleteMetadata" operation is:
List of documents successfully deleted:
/mypackage/jobs.xml
/mypackage/mo.xml
/mypackage/mdssys/cust/site/site1/jobs.xml.xml
/mypackage/mdssys/cust/site/site1/mo.xml.xml
4 documents successfully deleted.
```

The following example deletes metadata files under the package `mypackage` from `mdsApp` deployed in the server `server1` and excludes extended metadata and all customizations:

```
wls:/weblogic/serverConfig> deleteMetadata(application='mdsapp',
        server='server1', 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.
```

The following example deletes metadata files belonging to tenant `tenant1` under the package `mypackage` from the application `app1` deployed in the server `server1`:

```
wls:/weblogic/serverConfig> deleteMetadata(application='app1', server='server1',
    docs='/mypackage/**', tenantName='tenant1')
Executing operation: deleteMetadata.
deleteMetadata" operation completed. Summary of "deleteMetadata" operation is:
List of documents successfully deleted:
/mypackage/jobs.xml
/mypackage/mdssys/cust/site/site1/jobs.xml.xml
/mypackage/mdssys/cust/site/site2/mo.xml.xml
/mypackage/mdssys/cust/user/user1/mo.xml.xml
```

8.2.2 exportMetadata

Command Category: Application Metadata

Use with WLST: Online

8.2.2.1 Description

Exports application metadata. Use this command and the importMetadata command to transfer application metadata from one server location (for example, testing) to another server location (for example, production).

This command exports application metadata including customizations. However, by default, only those customizations from customization classes that are defined in the cust-config element of adf.config.xml are exported. To export customizations from customization classes not declared, use the restrictCustTo option.

8.2.2.2 Syntax

```
exportMetadata(application, server, toLocation  [, docs]
  [, restrictCustTo] [, excludeCustFor] [, excludeAllCust] [, excludeBaseDocs]
  [, excludeExtendedMetadata] [, excludeSeededDocs]
  [, fromLabel][, toLabel] [, applicationVersion] [, remote] [, tenantName])
```

Argument	Definition
application	The name of the application from which the metadata is to be exported.
server	The target server on which this application is deployed.
toLocation	The target directory or archive file (.jar, .JAR, .zip or .ZIP) to which documents selected from the source partition will be transferred. If you export to a directory, the directory must be a local or network directory or file where the application is physically deployed. If you export to an archive, the archive can be located on a local or network directory or file where the application is physically deployed, or on the system on which you are executing the command. If the location does not exist in the file system, a directory will be created except that when the names ends with .jar, .JAR, .zip or .ZIP, an archive file will be created. If the archive file already exists, the exportMetadata operation will overwrite the file. This argument can be used as temporary file system for transferring metadata from one server to another. For more information, see "Moving Metadata from a Test System to a Production System" in the *Oracle Fusion Middleware Administrator's Guide*.

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 "/**", which exports all the metadata in the repository. The asterisk (*) represents all documents under the current namespace. The double asterisk (**) represents all documents under the current namespace and also recursively includes all documents in subnamespaces. 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. Valid values are percent (%) or 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. Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas.
	
	For example:
	
	`restrictCustTo="user[scott]"`
	`restrictCustTo="site[site1],user[scott]"`
	`restrictCustTo="site[site1, %_2],user[scott, m%]"`
	
	If you do not specify this argument, only customization classes declared in the cust-config element of adf-config.xml are exported. If there is no cust-config element declared in adf-config.xml, all customization classes are exported.
	
	If you specify percent (%) as the value of this argument, all customizations are exported, whether or not they are declared in the cust-config element of adf-config.xml.
	
	Use this option to export all customizations or a subset of declared customizations. You can also use this option to export customizations from customization classes that are not declared in the cust-config element of adf-config.xml.
	
	This argument is ignored if the excludeAllCust argument is also specified.

excludeCustFor	Optional. A list of comma-separated customization layer names used to restrict the export operation to exclude customization documents that match the specified customization layers from being exported. This argument is 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 and excludeCustFor arguments.
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.

Argument	Definition
excludeSeededDocs	Optional. A Boolean value (true or false) that specifies whether all documents or only non-seeded documents are exported. Seeded documents are those documents that are packaged in a MAR. To exclude seeded documents, specify true. The default is 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.
remote	Optional. A Boolean value (true or false) that specifies whether the archive file will be written to a location where the application is deployed (false) or to the system on which you are executing the command (true). The default is false.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.2.2.3 Examples

The following example exports all metadata files from the application `mdsapp` deployed in the server `server1`.

```
wls:/weblogic/serverConfig> exportMetadata(application='mdsapp',
                server='server1',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
```

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='server1',toLocation='/tmp/myrepos',
                        restrictCustTo='user',
                        excludeBaseDocs='true',
                        fromLabel='label1',
                        toLabel='label2',
                        applicationVersion='11.1.1')
List of documents successfully transferred:
/mypackage/mdssys/cust/user/user1/write1.xml.xml
/mypackage/mdssys/cust/user/user2/write2.xml.xml
2 documents successfully transferred.
```

8.2.3 importMetadata

Command Category: Application Metadata

Use with WLST: Online

8.2.3.1 Description

Imports application metadata. Use the exportMetadata command and this command to transfer application metadata from one server location (for example, testing) to another server location (for example, production).

8.2.3.2 Syntax

```
importMetadata(application, server, fromLocation  [, docs]
 [, restrictCustTo] [, excludeAllCust] [, excludeBaseDocs]
 [, excludeExtendedMetadata] [, excludeUnmodifiedDocs]
  [, cancelOnException] [, applicationVersion] [, remote] [, tenantName])
```

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 or archive file from which documents will be selected for transfer. If you exported to a directory, the directory must be a local or network directory or file where the application is physically deployed. If you exported to an archive, the archive can be located on a local or network directory or file where the application is physically deployed, or on the system on which you are executing the command. This argument can be used as a temporary file system location for transferring metadata from one server to another. For more information, see "Moving Metadata from a Test System to a Production System" in the *Oracle Fusion Middleware Administrator's Guide*
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 "/**", which imports all of the documents in the repository. The asterisk (*) represents all documents under the current namespace. The double asterisk (**) represents all documents under the current namespace and also recursively includes all documents in subnamespaces. 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. Valid values are percent (%) or 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, including customization classes that are not declared in the cust-config element of adf-config.xml. Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas.
	For example:
	`restrictCustTo="user[scott]"`
	`restrictCustTo="site[site1],user[scott]"`
	`restrictCustTo="site[site1, %_2],user[scott, m%]"`
	If you do not specify this argument, only customization classes declared in the cust-config element of adf-config.xml are imported. If there is no cust-config element declared in adf-config.xml, all customization classes are imported.
	If you specify percent (%) as the value of this argument, all customizations are imported, whether or not they are declared in the cust-config element of adf-config.xml.
	Use this option to import all customizations or a subset of declared customizations. You can also use this option to export customizations from customization classes that are not declared in the cust-config element of adf-config.xml.
	This argument is 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.
excludeUnmodifiedDocs	Optional. A Boolean value (true or false) that specifies whether only changed documents are imported. If you specify true, only changed documents are imported. The default is false.
cancelOnException	Optional. A Boolean value (true or false) that specifies whether or not to abort the import operation when an exception is encountered. The default is true.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.
remote	Optional. A Boolean value (true or false) that specifies whether the archive file is in a location where the application is deployed (false) or on the system on which you are executing the command (true). The default is false.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.2.3.3 Example

The following example imports all metadata available in /tmp/myrepos to the application mdsapp deployed in the server server1:

```
wls:/weblogic/serverConfig> importMetadata(application='mdsapp', server='server1',
                            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.
```

8.2.4 purgeMetadata

Command Category: Application Metadata

Use with WLST: Online

8.2.4.1 Description

Purges the older (non-tip) versions of unlabeled documents from the application's repository. All unlabeled documents will be purged if they are expired, based on Time-To-Live (the olderThan argument). This command is applicable only for repositories that support versioning, that is, database-based repositories.

8.2.4.2 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.
olderThan	Document versions that are older than this value (in seconds) will be purged.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

8.2.4.3 Example

The following example purges the document version history for the application mdsapp deployed in the server server1, if the version is older than 10 seconds:

```
wls:/weblogic/serverConfig> purgeMetadata('mdsapp', 'server1', 10)
Executing operation: purgeMetadata.
Metadata purged: Total number of versions: 10.
Number of versions purged: 0.
```

8.3 Sandbox Metadata Management Commands

Use the commands in Table 8–4 to manage metadata in a sandbox. A sandbox is a temporary location for testing changes before moving them to a production system. Sandboxes are not visible to most users until they are applied.

Table 8–4 Sandbox Metadata Management Commands

Use this command...	To...	Use with WLST...
exportSandboxMetadata	Exports the metadata from a sandbox.	Online
importSandboxMetadata	Imports metadata into a sandbox.	Online

8.3.1 exportSandboxMetadata

Command Category: Sandbox Metadata Management

Use with WLST: Online

8.3.1.1 Description

Exports the changes to the metadata from a sandbox on a test system.

You can only use this command with a database-based MDS repository.

8.3.1.2 Syntax

```
exportSandboxMetadata(application, server, toArchive, sandboxName
          [, restrictCustTo] [, applicationVersion] [, remote] [, tenantName)
```

Argument	Definition
application	The name of the application from which the metadata is to be exported.
server	The target server on which this application is deployed.
toArchive	The target archive file (.jar, .JAR, .zip or .ZIP) to which the sandbox contents will be transferred. The archive can be located on a local or network directory where the application is physically deployed. If you specify the -remote argument, the archive can be located on the system on which you are executing the command.
sandboxName	The name of the sandbox to export.

restrictCustTo	Optional. Valid values are percent (%) or 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. Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas.
	For example:
	`restrictCustTo="user[scott]"`
	`restrictCustTo="site[site1],user[scott]"`
	`restrictCustTo="site[site1, %_2],user[scott, m%]"`
	If you do not specify this argument or if you specify percent (%) as the value of this argument, all customizations are exported, whether or not they are declared in the cust-config element of adf-config.xml.
	Use this option to export all customizations or a subset of declared customizations. You can also use this option to export customizations from customization classes that are not declared in the cust-config element of adf-config.xml.
	This argument is ignored if the excludeAllCust argument is also specified.

Argument	Definition
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.
remote	Optional. A Boolean value (true or false) that specifies whether the archive file will be written to a location where the application is deployed (false) or to the system on which you are executing the command (true). The default is false.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.3.1.3 Example

The following example exports a sandbox from the MDS repository for the application myapp:

```
wls:/weblogic/serverConfig>exportSandboxMetadata('myapp', 'server1',
                    '/tmp/sandbox1.jar', 'sandbox1')
```

8.3.2 importSandboxMetadata

Command Category: Sandbox Metadata Management

Use with WLST: Online

8.3.2.1 Description

Imports the contents of a sandbox archive to another sandbox in the MDS repository partition of the specified application. It can also update the contents of a given archive to a sandbox in the MDS repository partition of a given application. All customizations are imported, whether or not they are declared in the cust-config element of adf-config.xml.

You can only use this command with a database-based MDS repository.

8.3.2.2 Syntax

```
importSandboxMetadata(application, server, fromArchive  [, forceSBCreation]
               [, useExistingSandbox] [, sandboxName] [, applicationVersion]
               [, remote] [, tenantName])
```

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.
fromArchive	The source archive file from which documents will be selected for transfer. The archive can be located on a local or network directory where the application is physically deployed. If you specify the `-remote` argument, the archive can be located on the system on which you are executing the command.

Argument	Definition
forceSBCreation	Optional. A Boolean value (true or false) that specifies whether the operation will overwrite an existing sandbox with the same name. When the argument is set to `true`, if the fromArchive argument specifies a sandbox with the same name as one that already exists in the application's partition, the original sandbox is deleted and a new sandbox is created. When the argument is set to `false`, if a sandbox with the same name exists, an exception is thrown. The default is `false`.
useExistingSandbox	Optional. When set to **true**, the contents of the archive are imported to the sandbox specified with the `sandboxName` argument. This argument is ignored if there is no value specified for `sandboxName`. The default is `false`.
sandboxName	Optional. The name of the sandbox to update. This argument is ignored if `useExistingSandbox` is `false`.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.
remote	Optional. A Boolean value (true or false) that specifies whether the archive file is in a location where the application is deployed (false) or on the system on which you are executing the command (true). The default is `false`.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.3.2.3 Examples

The following example imports the contents of the sandbox sandbox1.jar:

```
wls:/weblogic/serverConfig> importSandboxMetadata(application='myapp', 'server1'',
                            '/tmp/sandbox1.jar')
```

The following example updates the sandbox sandbox1.jar:

```
wls:/weblogic/serverConfig>importSandboxMetadata('myapp', 'server1',
'/tmp/sandbox1.jar', useExistingSandbox='true', sandboxName='sandbox1')
```

8.4 Application Label Management Commands

Use the commands in Table 8–5 to manage labels for applications.

Table 8–5 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
purgeMetadataLabels	Deletes the labels matching the specified criteria.	Online

8.4.1 createMetadataLabel

Command Category: Application Label Management

Use with WLST: Online

8.4.1.1 Description

Creates a new label for the documents in the application's repository partition. This command is applicable only for repositories that support versioning.

8.4.1.2 Syntax

```
createMetadataLabel(application, server, name [, applicationVersion] [,
tenantName])
```

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. If the application is deployed to multiple Managed Servers in a cluster, you can use the name of any of the server names. You cannot specify multiple server names.
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.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.4.1.3 Example

The following example creates the label `label1` for the application `mdsapp` deployed in the server `server1`:

```
wls:/weblogic/serverConfig> createMetadataLabel('mdsapp','server1','label1')
Executing operation: createMetadataLabel.
Created metadata label "label1".
```

8.4.2 deleteMetadataLabel

Command Category: Application Label Management

Use with WLST: Online

8.4.2.1 Description

Deletes a label for the documents in the application's repository partition. This command is applicable only for repositories that support versioning.

8.4.2.2 Syntax

```
deleteMetadataLabel(application, server, name [, applicationVersion] [,
tenantName])
```

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. If the application is deployed to multiple Managed Servers in a cluster, you can use the name of any of the server names. You cannot specify multiple server names.
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.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.4.2.3 Example

The following example deletes the metadata label `label1` from the application `mdsapp` deployed in the server `server1`:

```
wls:/weblogic/serverConfig> deleteMetadataLabel('mdsapp','server1','label1')
Executing operation: deleteMetadataLabel.
Deleted metadata label "label1".
```

8.4.3 listMetadataLabels

Command Category:

Use with WLST: Online

8.4.3.1 Description

Lists all of the metadata labels in the application's repository partition. This command is applicable only for repositories that support versioning.

8.4.3.2 Syntax

```
listMetadataLabels(application, server [, applicationVersion] [, tenantName])
```

Argument	Definition
application	The name of the application for which all of the labels in the repository partition should be listed.
server	The target server on which this application is deployed. If the application is deployed to multiple Managed Servers in a cluster, you can use the name of any of the server names. You cannot specify multiple server names.
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.4.3.3 Example

The following example lists the metadata labels available for the application `mdsapp` deployed in the server `server1`:

```
wls:/weblogic/serverConfig> listMetadataLabels('mdsapp', 'server1')
Executing operation: listMetadataLabels.
Database Repository partition contains the following labels:
label2
label3
```

8.4.4 promoteMetadataLabel

Command Category: Application Label Management

Use with WLST: Online

8.4.4.1 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.

8.4.4.2 Syntax

```
promoteMetadataLabel(application, server, name [, applicationVersion] [,
tenantName])
```

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. If the application is deployed to multiple Managed Servers in a cluster, you can use the name of any of the server names. You cannot specify multiple server names.
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.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.4.4.3 Example

The following example promotes the metadata label `label1` to tip in the application `mdsapp` deployed in the server `server1`:

```
wls:/weblogic/serverConfig> promoteMetadataLabel('mdsapp', 'server1','label1')
Executing operation: promoteMetadataLabel.
Promoted metadata label "label1" to tip.
```

8.4.5 purgeMetadataLabels

Command Category: Application Label Management

Use with WLST: Online

8.4.5.1 Description

Purges or lists the metadata labels that match the given pattern or age, but does not delete the metadata documents that were part of the label. You can delete the documents by executing the purgeMetadata command.

8.4.5.2 Syntax

```
purgeMetadataLabels(repository, partition [, namePattern] [, olderThanInMin]
 [, infoOnly] [, tenantName])
```

Argument	Definition
repository	The name of the MDS repository that contains the partition whose metadata labels will be purged or listed.
partition	The name of the partition whose metadata labels will be purged or listed.
namePattern	Optional. A pattern that matches the names of labels. The pattern can contain the following special characters: ■ The percent (%) character, which matches any number of characters. ■ The underscore (_) character, which matches exactly one arbitrary character. ■ The backslash character ('\'), which can be used to escape the percent, the underscore, and the backslash (itself) characters, so they match only %, _, or \.
olderThanInMin	Optional. The age of the labels, in minutes. The default is 525600 (one year).
infoOnly	Optional. Valid values are true or false. If you set it to `true`, it does not purge the labels, but lists the labels that match the specified pattern. The default is `false`.
tenantName	A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored.

8.4.5.3 Examples

The following example lists the labels that match the specified namePattern, but does not delete them:

```
wls:/weblogic/serverConfig> purgeMetadataLabels(repository='mds-myRepos',
          partition='partition1', namePattern='mylabel*', infoOnly='true' )
```

The following example purges the labels that match the specified namePattern and that are older than a year:

```
wls:/weblogic/serverConfig> purgeMetadataLabels(repository='mds-myRepos',
                   partition='partition1', namePattern='mylabel*')
```

The following example deletes labels that match the specified namePattern and that are older than 30 minutes:

```
wls:/weblogic/serverConfig> purgeMetadataLabels(repository='mds-myRepos',
                  partition='partition1',
                 namePattern='mylabel*', olderThanInMin='30')
```

8.5 Application Management Deployment Commands

Use the commands in Table 8–6 to manage deployment.

Table 8–6 Application Management Deployment Commands

Use this command...	To...	Use with WLST...
getMDSArchiveConfig	Returns an MDSArchiveConfig object.	Offline
importMAR	Imports an MAR.	Online

8.5.1 getMDSArchiveConfig

Command Category: Application Management Deployment

Use with WLST: Offline

8.5.1.1 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 contain 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 contain 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 argument, 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.

8.5.1.2 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 is not considered if the type is set to File.
path	Optional. The directory for the metadata files. This argument is required if the type is set to File. This argument is not 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, including 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.

8.5.1.3 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')
```

8.5.2 importMAR

Command Category: Application Management Deployment

Use with WLST: Online

8.5.2.1 Description

Imports the metadata from the MAR that is packaged with the application's EAR file. If the MAR had already been imported into the partition, the command deletes the previous version and imports the new version.

8.5.2.2 Syntax

```
importMAR(application, server [, force] [, 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.
force	Optional. A Boolean value (true or false) that specifies whether only changed documents and MARs are imported. For a database-based repository, if you set this argument to false, only new or changed documents from changed MARs are imported. The command creates a label for each MAR for which documents are imported. The label has the following format: postDeploy_*application_name_MAR_name_MAR_checksum* For a file-based repository, if you set this argument to false, only changed MARs are imported. The command does not compare individual documents The command creates a file in the repository for each imported MAR. The default is true.

Argument	Definition
applicationVersion	Optional. The application version, if multiple versions of the same application are deployed.

8.5.2.3 Example

The following example imports metadata from the MAR to the application `mdsapp`:

```
wls:/weblogic/serverConfig> importMAR('mdsapp','server1')
Executing operation: importMAR.
"importMAR" operation completed. Summary of "importMAR" operation is:
/app1/jobs.xml
/app1/mo.xml
2 documents successfully transferred.
```

8.6 Multitenancy Management Commands

Use the commands in Table 8–7 to manage tenants.

Table 8–7 Multitenancy Management Commands

Use this command...	To...	Use with WLST...
deprovisionTenant	Deprovisions a tenant from the metadata store.	Online
listTenants	Lists the tenants.	Online

8.6.1 deprovisionTenant

Deprovisions a tenant from the metadata store. All metadata associated with the tenant will be removed from the store

8.6.1.1 Syntax

```
deprovisionTenant(repository, partition, tenantName)
```

Argument	Definition
repository	The name of the repository that contains the tenant.
partition	The name of the partition that contains the tenant.
tenantName	A unique name identifying the tenant to use for this operation.

8.6.1.2 Example

The following example deprovisions the tenant with tenantName `tenant1`:

```
wls:/weblogic/serverConfig> deprovisionTenant("mds-myrepos", "part1", "tenant1")
Executing operation: deprovisionTenant.
Tenant "tenant1" has been deprovisioned.
```

8.6.2 listTenants

Lists all tenants in an MDS Repository partition.

8.6.2.1 Syntax

```
listTenants(repository, partition)
```

Argument	Definition
repository	The name of the repository that contains the tenants.
partition	The name of the partition that contains the tenants.

8.6.2.2 Example

The following example lists all tenants in the specified repository and partition:

```
wls:/weblogic/serverConfig> listTenants("mds-myrepos", "part1")
Executing operation: listTenants.
0  GLOBAL
1  tenant1
2  tenant2
3  tenant3
```

9

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*.

9.1 Overview of WSLT Command Categories

WLST commands are divided into the categories shown in Table 9–1.

Table 9–1 Oracle SOA Suite Command Categories

Command category	Description
Deployment Commands	Deploy and undeploy SOA composite applications.

Table 9–1 (Cont.) Oracle SOA Suite Command Categories

Command category	Description
SOA Composite Application Management Commands	Start, stop, activate, retire, assign a default revision version, and list deployed SOA composite applications.
Configuration Plan Management Commands	Attach, extract, generate, and validate configuration plans for SOA composite applications.
Task Validation Commands	Validate human workflow tasks.
SOA Composite Application Compilation Commands	Compile SOA composite applications.
SOA Composite Application Packaging Commands	Package SOA composite applications into archive files to deploy.
SOA Composite Application Test Commands	Test SOA composite applications prior to deployment in a production environment.
SOA Composite Application HTTP Client-Based Export and Import Commands	Export and import SOA composite applications based on the HTTP client.
SOA Composite Application MBean-Based Export and Import Commands	Export and import SOA composite applications on the server-based composite store MBean (`CompositeStoreMXBean`).
SOA Composite Application Partition Management Commands	Logically group different revisions of your SOA composite applications into separate sections.

9.2 Deployment Commands

Use the deployment commands, listed in Table 9–2, to deploy and undeploy SOA composite applications.

Table 9–2 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

9.2.1 sca_deployComposite

Command Category: Deployment Commands

Use with WLST: Offline

9.2.1.1 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 9.7,

"SOA Composite Application Packaging Commands" for instructions on packaging a SOA composite application.

9.2.1.2 Syntax

```
sca_deployComposite(serverURL, sarLocation, [overwrite], [user], [password],
[forceDefault], [configplan], [partition])
```

Argument	Definition
serverURL	URL of the server that hosts the SOA Infrastructure application (for example, `http://myhost10:7001`).
sarLocation	Absolute path to one the following: ■ SOA archive (SAR) file. A SAR file is a special JAR file that requires a prefix of `sca_` (for example, `sca_HelloWorld_rev1.0.jar`). The SAR file can be deployed with the deployment commands (such as `sca_deployComposite()`), but a regular `.jar` file is not treated as a special 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	Optional. Absolute path of a configuration plan to be applied to a specified SAR file or to all SAR files included in the ZIP file.
partition	Optional. The name of the partition in which to deploy the SOA composite application. The default value is `default`. If you do not specify a partition, the composite is automatically deployed into the `default` partition.

Note: Human workflow artifacts such as task mapped attributes (previously known as flex field mappings) and rules (such as vacation rules) are defined based on the namespace of the task definition. Therefore, the following issues are true when the same SOA composite application with a human workflow task is deployed into multiple partitions:

- For the same task definition type, mapped attributes defined in one partition are visible in another partition.
- Rules defined on a task definition in one partition can apply to the same definition in another partition.

9.2.1.3 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")
```

The following example deploys the `HelloWorld` application to the `myPartition` partition.

```
wls:/mydomain/ServerConfig> sca_deployComposite("http://stadp10:7001",
"/tmp/sca_HelloWorld_rev1.0.jar", partition="myPartition")
```

9.2.2 sca_undeployComposite

Command Category: Deployment Commands

Use with WLST: Offline

9.2.2.1 Description

Undeploys a currently deployed SOA composite application.

9.2.2.2 Syntax

```
sca_undeployComposite(serverURL, compositeName, revision, [user], [password],
[partition])
```

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.

Argument	Definition
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.
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`. If you do not specify a partition, the `default` partition is searched for the SOA composite application. However, no other partitions are searched.

9.2.2.3 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:
```

The following example undeploys the `HelloWorld` application in the `myPartition` partition.

```
wls:/mydomain/ServerConfig> sca_undeployComposite("http://stadp10:7001",
 "HelloWorld", "1.0", partition='myPartition')
```

9.3 SOA Composite Application Management Commands

Use the management commands, listed in Table 9–3, to start, stop, activate, retire, assign a default revision version, and list deployed SOA composite applications.

Table 9–3 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
sca_assignDefaultComposite	Assign the default revision version to a SOA composite application.	Offline
sca_getDefaultCompositeRevision	List the revision of the default composite of the given composite series.	Offline
sca_listDeployedComposites	List the deployed SOA composite applications.	Offline

9.3.1 sca_startComposite

Command Category: Application Management Commands

Use with WLST: Offline

9.3.1.1 Description

Starts a previously stopped SOA composite application.

9.3.1.2 Syntax

```
sca_startComposite(host, port, user, password, compositeName, revision, [label],
[partition])
```

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.
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`. If you do not specify a partition, the `default` partition is searched for the SOA composite application. However, no other partitions are searched.

9.3.1.3 Example

The following example starts revision `1.0` of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_startComposite("myhost", "7001", "weblogic",
"welcome1", "HelloWorld", "1.0")
```

The following example starts revision `1.0` of the `HelloWorld` application in the partition `myPartition`.

```
wls:/mydomain/ServerConfig> sca_startComposite("stadp10", "7001", "weblogic",
 "weblogic", "HelloWorld", "1.0", partition="myPartition")
```

9.3.2 sca_stopComposite

Command Category: Application Management Commands

Use with WLST: Offline

9.3.2.1 Description

Stops a currently running SOA composite application.

9.3.2.2 Syntax

```
sca_stopComposite(host, port, user, password, compositeName, revision, [label],
 [partition])
```

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.
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`. If you do not specify a partition, the `default` partition is searched for the SOA composite application. However, no other partitions are searched.

9.3.2.3 Example

The following example stops revision `1.0` of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_stopComposite("myhost", "7001", "weblogic",
"welcome1", "HelloWorld", "1.0")
```

The following example stops revision `1.0` of the `HelloWorld` application in the partition `myPartition`.

```
wls:/mydomain/ServerConfig> sca_stopComposite("stadp10", "7001", "weblogic",
 "weblogic", "HelloWorld", "1.0", partition="myPartition")
```

9.3.3 sca_activateComposite

Command Category: Application Management Commands

Use with WLST: Offline

9.3.3.1 Description

Activates a retired SOA composite application and its instances. You can then create new instances.

9.3.3.2 Syntax

```
sca_activateComposite(host, port, user, password, compositeName, revision,
[label], [partition])
```

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.

Argument	Definition
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.
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`. If you do not specify a partition, the `default` partition is searched for the SOA composite application. However, no other partitions are searched.

9.3.3.3 Example

The following example activates revision `1.0` of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_activateComposite("myhost", "7001", "weblogic",
"welcome1", "HelloWorld", "1.0")
```

The following example activates revision `1.0` of the `HelloWorld` application in the partition `myPartition`.

```
wls:/mydomain/ServerConfig> sca_activateComposite("stadp10", "7001", "weblogic",
 "weblogic", "HelloWorld", "1.0", partition="myPartition")
```

9.3.4 sca_retireComposite

Command Category: Application Management Commands

Use with WLST: Offline

9.3.4.1 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.

9.3.4.2 Syntax

```
sca_retireComposite(host, port, user, password, compositeName, revision, [label],
[partition])
```

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.
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`. If you do not specify a partition, the `default` partition is searched for the SOA composite application. However, no other partitions are searched.

9.3.4.3 Example

The following example retires revision `1.0` of the `HelloWorld` application.

```
wls:/mydomain/ServerConfig> sca_retireComposite("myhost", "7001", "weblogic",
"welcome1", "HelloWorld", "1.0")
```

The following example retires revision `1.0` of the `HelloWorld` application in the partition `myPartition`.

```
wls:/mydomain/ServerConfig> sca_retireComposite("stadp10", "7001", "weblogic",
 "weblogic", "HelloWorld", "1.0", partition="myPartition")
```

9.3.5 sca_assignDefaultComposite

Command Category: Application Management Commands

Use with WLST: Offline

9.3.5.1 Description

Sets a SOA composite application revision as the default version. This revision is instantiated when a new request comes in.

9.3.5.2 Syntax

```
sca_assignDefaultComposite(host, port, user, password, compositeName, revision,
[partition])
```

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.
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`. If you do not specify a partition, the `default` partition is searched for the SOA composite application. However, no other partitions are searched.

9.3.5.3 Example

The following example sets revision `1.0` of the `HelloWorld` application as the default version.

```
wls:/mydomain/ServerConfig> sca_assignDefaultComposite("myhost", "7001",
"weblogic", "welcome1", "HelloWorld", "1.0")
```

The following example sets revision `1.0` of the `HelloWorld` application located in the partition `myPartition` as the default version.

```
wls:/mydomain/ServerConfig> sca_assignDefaultComposite("stadp10", "7001",
 "weblogic", "weblogic", "HelloWorld", "1.0", partition="myPartition")
```

9.3.6 sca_getDefaultCompositeRevision

Command Category: Application Management Commands

Use with WLST: Offline

9.3.6.1 Description

Lists the revision of the default composite of the given composite series.

9.3.6.2 Syntax

```
sca_getDefaultCompositeRevision(host, port, user, password, compositeName,
partition)
```

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.
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`. If you do not specify a partition, the default partition is searched for the SOA composite application. However, no other partitions are searched.

9.3.6.3 Example

The following example returns the revision of the default composite of the given composite series.

```
wls:/mydomain/ServerConfig> sca_getDefaultCompositeRevision("myhost", "7001",
"weblogic", "weblogic","HelloWorld")
```

The following example returns the revision of the default composite of the given composite series in the partition named `myPartition`.

```
wls:/mydomain/ServerConfig> sca_getDefaultCompositeRevision("myhost", "7001",
"weblogic", "weblogic", "HelloWorld", partition="myPartition")
```

9.3.7 sca_listDeployedComposites

Command Category: Application Management Commands

Use with WLST: Offline

9.3.7.1 Description

Lists all SOA composite applications deployed to the SOA platform.

9.3.7.2 Syntax

```
sca_listDeployedComposites(host, port, user, password)
```

Argument	Definition
host	Hostname of the Oracle WebLogic Server (for example, `myhost`).

Argument	Definition
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.

9.3.7.3 Example

The following example lists all the deployed SOA composite applications on the server myhost.

```
wls:/mydomain/ServerConfig> sca_listDeployedComposites('myhost', '7001',
'weblogic', 'welcome1')
```

9.4 Configuration Plan Management Commands

Use the configuration plan management commands, listed in Table 9–4, to attach, extract, generate, and validate configuration plans for SOA composite applications.

Table 9–4 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
sca_generatePlan	Generate a configuration plan for editing.	Offline
sca_validatePlan	Validate the configuration plan.	Offline

9.4.1 sca_attachPlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

9.4.1.1 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.

9.4.1.2 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.

Argument	Definition
verbose	Optional. Indicates whether to print more information about the configuration plan attachment. ■ `true` (default): Prints more information. ■ `false`: Does not print more information.

9.4.1.3 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)
```

9.4.2 sca_extractPlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

9.4.2.1 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`.

9.4.2.2 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.

9.4.2.3 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)
```

9.4.3 sca_generatePlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

9.4.3.1 Description

Generates a configuration plan for editing.

9.4.3.2 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.
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.

9.4.3.3 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)
```

9.4.4 sca_validatePlan

Command Category: Configuration Plan Management Commands

Use with WLST: Offline

9.4.4.1 Description

Validates the configuration plan. This command identifies all search and replacement changes to be made during deployment. Use this option for debugging only.

9.4.4.2 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.

9.4.4.3 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)
```

9.5 Task Validation Commands

Use the task validation command, listed in Table 9–5, to validate human workflow tasks.

Table 9–5 Task Validation Command for WLST Configuration

Use this command...	To...	Use with WLST...
sca_validateTask	Validate a human workflow task.	Offline

9.5.1 sca_validateTask

Command Category: Task Validation Commands

Use with WLST: Offline

9.5.1.1 Description

Validates a human workflow task contained in the `.task` file that you created when designing a human task in the Human Task Editor.

9.5.1.2 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`.

9.5.1.3 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)
```

9.6 SOA Composite Application Compilation Commands

Use the compilation commands, listed in Table 9–6, to compile SOA composite applications.

Table 9–6 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

9.6.1 sca_setProp

Command Category: Application Compilation Commands

Use with WLST: Offline

9.6.1.1 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.

9.6.1.2 Syntax

```
sca_setProp(propName, propValue)
```

Argument	Definition
propName	Property name.
propValue	Property value.

9.6.1.3 Example

The following example sets the property name and property value.

```
wls:/mydomain/ServerConfig> sca_setProp("oracle.home",
"/scratch/myusername/beahome/AS11gR1SOA")
```

9.6.2 sca_compile

Command Category: Application Compilation Commands

Use with WLST: Offline

9.6.2.1 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.

9.6.2.2 Syntax

```
sca_compile(composite, [outXml], [error], [appHome], [displayLevel], [oracleHome])
```

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.

9.6.2.3 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")
```

9.7 SOA Composite Application Packaging Commands

Use the packaging command, listed in Table 9–7, to package SOA composite applications into a composite SAR file.

Table 9–7 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

9.7.1 sca_package

Command Category: Application Packaging Commands

Use with WLST: Offline

9.7.1.1 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.

9.7.1.2 Syntax

```
sca_package(compositeDir, compositeName, revision, [appHome], [oracleHome])
```

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.

9.7.1.3 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")
```

9.8 SOA Composite Application Test Commands

Use the SOA composite application test command, listed in Table 9–8, to test a SOA composite applications.

Table 9–8 SOA Composite Application Test Command for WLST Configuration

Use this command...	To...	Use with WLST...
sca_test	Test deployed SOA composite applications.	Offline

9.8.1 sca_test

Command Category: Application Test Commands

Use with WLST: Offline

9.8.1.1 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.

9.8.1.2 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.

9.8.1.3 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')
```

9.9 SOA Composite Application HTTP Client-Based Export and Import Commands

Use the SOA composite application commands, listed in Table 9–9, to export and import SOA composite applications based on the HTTP client. The SOA Infrastructure must be running to use these commands.

Table 9–9 SOA Composite Application Export and Import Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_exportComposite	Export a SOA composite application into a SAR file.	Offline
sca_exportUpdates	Export postdeployment changes of a SOA composite application into a JAR file.	Offline
sca_importUpdates	Import postdeployment changes of a SOA composite application.	Offline
sca_exportSharedData	Export shared data of a given pattern into a JAR file.	Offline
sca_removeSharedData	Removes a top-level shared data folder.	Offline

9.9.1 sca_exportComposite

Command Category: Application Export and Import Commands

Use with WLST: Offline

9.9.1.1 Description

Exports a SOA composite application into a SAR file.

9.9.1.2 Syntax

```
sca_exportComposite(serverURL, updateType, sarFile, compositeName, revision,
[user], [password], [partition])
```

Argument	Definition
serverURL	URL of the server that hosts the SOA Infrastructure application (for example, `http://stabc:8001`).
updateType	Type of postdeployment changes to be exported: ■ `all`: Includes all postdeployment changes. ■ `property`: Includes only property postdeployment changes (binding component properties, composite properties such as audit level settings and payload validation status, and policy attachments). ■ `runtime`: Includes only runtime (rules dictionary and domain value maps (DVMs)) and metadata postdeployment changes. ■ `none`: Exports the original composite without any postdeployment changes (including property changes and runtime changes).
sarFile	Absolute path of a SAR file to generate (a `.jar` file that begins with `sca_`).
compositeName	Name of the composite to export.
revision	Revision of the composite to export.

Argument	Definition
user	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: `user='username'`
password	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: `password='password'`
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`.

9.9.1.3 Examples

The following example exports the composite without including any postdeployment changes.

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001',
'none', '/tmp/sca_HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

The following example exports a composite with all postdeployment updates.

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001',
'all', '/tmp/sca_HelloWorld_ rev1.0-all.jar', 'HelloWorld', '1.0')
```

The following example exports a composite with property postdeployment updates.

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001',
'property', '/tmp/sca_HelloWorld_ rev1.0-prop.jar', 'HelloWorld', '1.0')
```

The following example exports a composite with runtime/metadata postdeployment updates.

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001',
'runtime', '/tmp/sca_HelloWorld_ rev1.0-runtime.jar', 'HelloWorld', '1.0')
```

The following example exports a composite in the `myPartition` partition without including any postdeployment updates:

```
wls:/offline/mydomain/ServerConfig> sca_exportComposite('http://stabc:8001',
 'none', '/tmp/sca_HelloWorld_rev1.0.jar', 'HelloWorld', '1.0',
 partition='myPartition')
```

9.9.2 sca_exportUpdates

Command Category: Application Export and Import Commands

Use with WLST: Offline

9.9.2.1 Description

Exports postdeployment changes of a SOA composite application into a JAR file.

9.9.2.2 Syntax

```
sca_exportUpdates(serverURL, updateType, jarFile, compositeName, revision,
[user], [password], [partition])
```

Argument	Definition
serverURL	URL of the server that hosts the SOA Infrastructure application (for example, `http://stabc:8001`).
updateType	The type of postdeployment changes to be exported. ■ `all`: Includes all postdeployment changes. ■ `property`: Includes only property postdeployment changes (binding component properties, composite properties such as audit level settings and payload validation status, and policy attachments). ■ `runtime`: Includes only runtime (rules dictionary and domain value maps (DVMs)) and metadata postdeployment changes.
jarFile	Absolute path of a JAR file to generate. `sca_exportUpdates()` creates a regular `.jar` file that cannot be imported using regular deployment commands. It must be imported by using `sca_importUpdates()`.
compositeName	Name of the composite to export.
revision	Revision of the composite to export.
user	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: `user='`*username*`'`
password	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: `password='`*password*`'`
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`.

9.9.2.3 Examples

The following example exports all postdeployment updates.

```
wls:/offline/mydomain/ServerConfig>  sca_exportUpdates('http://stabc:8001', 'all',
'/tmp/all-HelloWorld_rev1.0.jar','HelloWorld', '1.0')
```

The following example exports property postdeployment updates.

```
wls:/offline/mydomain/ServerConfig> sca_exportUpdates('http://stabc:8001',
'property','/tmp/prop-HelloWorld_ rev1.0.jar', 'HelloWorld', '1.0')
```

The following example exports runtime/metadata postdeployment updates.

```
wls:/offline/mydomain/ServerConfig> sca_exportUpdates('http://stabc:8001',
'runtime','/tmp/runtime-HelloWorld_ rev1.0.jar', 'HelloWorld', '1.0')
```

The following example exports postdeployment changes of a composite in the partition `myPartition` into a JAR file.

```
wls:/offline/mydomain/ServerConfig> sca_exportUpdates(serverURL, updateType,
 jarFile, compositeName, revision, user=None, password=None,
 partition='myPartition')
```

9.9.3 sca_importUpdates

Command Category: Application Export and Import Commands

Use with WLST: Offline

9.9.3.1 Description

Imports postdeployment changes of a SOA composite application.

9.9.3.2 Syntax

```
sca_importUpdates(serverURL, jarFile, compositeName, revision, [user],
[password])
```

Argument	Definition
serverURL	URL of the server that hosts the SOA Infrastructure application (for example, `http://stabc:8001`).
jarFile	Absolute path of a JAR file that contains postdeployment changes.
compositeName	Name of the composite to which the postdeployment changes are imported.
revision	Revision of the composite to which the postdeployment changes are imported.
user	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: `user='username'`
password	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: `password='password'`
partition	Optional. The name of the partition in which the SOA composite application is located. The default value is `default`.

9.9.3.3 Examples

The following example imports postdeployment changes of a SOA composite application.

```
wls:/offline/mydomain/ServerConfig> sca_importUpdates('http://stabc:8001',
'/tmp/all-HelloWorld_rev1.0.jar', 'HelloWorld', '1.0')
```

The following example imports postdeployment changes of a composite in the partition `myPartition`.

```
wls:/offline/mydomain/ServerConfig> sca_importUpdates(serverURL, jarFile,
 compositeName, revision, user=None, password=None, partition='myPartition')
```

9.9.4 sca_exportSharedData

Command Category: Application Export and Import Commands

Use with WLST: Offline

9.9.4.1 Description

Exports shared data of a given pattern into a JAR file.

9.9.4.2 Syntax

```
sca_exportSharedData(serverURL, jarFile, pattern, [user], [password])
```

Argument	Definition
serverURL	URL of the server that hosts the SOA Infrastructure application (for example, `http://stabc:8001`).
jarFile	Absolute path of a JAR file to generate.
pattern	The file pattern supported by MDS transfer APIs. Use the semicolon delimiter (`;`) if more than one pattern is specified. Exclude the shared data namespace `/apps` in the pattern. For example: `/Project1/**;/Project2/**` This example exports all documents under `/apps/Project1` and `/apps/Project2`.
user	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: `user='username'`
password	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: `password='password'`

9.9.4.3 Examples

The following example exports shared data of a given pattern into a JAR file.

```
wls:/offline/mydomain/ServerConfig> sca_exportSharedData('http://stabc:8001',
'/tmp/MySharedData.jar', '/Project1/**')
```

9.9.5 sca_removeSharedData

Command Category: Application Export and Import Commands

Use with WLST: Offline

9.9.5.1 Description

Removes a top-level shared data folder, even if there are composites deployed in the service engine.

9.9.5.2 Syntax

```
sca_removeSharedData(serverURL, folderName, [user], [password])
```

Argument	Definition
serverURL	URL of the server that hosts the SOA Infrastructure application (for example, `http://stabc:8001`).
folderName	The name of a top-level shared data folder to be removed.
user	Optional. The user name for accessing the server when basic configuration is configured. Use the following syntax for this argument: `user='username'`
password	Optional. The password for accessing the server when basic configuration is configured. Use the following syntax for this argument: `password='password'`

9.9.5.3 Examples

The following example removes the top-level shared data `Project1` folder.

```
sca_removeSharedData('http://stabc:8001', 'Project1')
```

9.10 SOA Composite Application MBean-Based Export and Import Commands

Use the deployment commands, listed in Table 9–10, to export and import SOA composite applications on the server-based composite store MBean (`CompositeStoreMXBean`).

Table 9–10 SOA Composite Application Export and Import Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_exportCompositeMb	Export a SOA composite application into a SAR file.	Online
sca_exportUpdatesMb	Export postdeployment changes of a SOA composite application into a JAR file.	Online
sca_importUpdatesMb	Import postdeployment changes of a SOA composite application.	Online
sca_exportSharedDataMb	Export shared data of a given pattern into a JAR file.	Online

If you use this option, note that the file generated in the export commands and the file read in the import command must be on the host where the server is running (either an Oracle WebLogic Administration Server or a managed SOA server).

The composite store MBean is registered as both a server runtime MBean of the SOA server and as a domain runtime MBean of the Oracle WebLogic Administration Server, which allows the import and export to continue working while SOA servers are down. Only WLST commands are provided for using the composite store MBean; there are no `ant` commands.

You must run the `connect()` command to connect to either a SOA server or an Oracle WebLogic Administration Server.

```
wls:offline>connect('weblogic', 'password', 't3://stabc:8001')
```

If you use the domain runtime MBean while the SOA servers are down, you must run the `domainRuntime()` command.

```
wls:offline>connect('weblogic', 'password', 't3://stabc:7001')
wls:/soainfra/serverConfig>domainRuntime()
```

9.10.1 sca_exportCompositeMb

Command Category: Application Export and Import Commands

Use with WLST: Online

9.10.1.1 Description

Exports a SOA composite application into a SAR file.

9.10.1.2 Syntax

```
sca_exportCompositeMb(updateType, sarFile, compositeName, revision)
```

Argument	Definition
updateType	Type of postdeployment changes to be exported: ■ `all`: All postdeployment changes are included. ■ `property`: Property changes are included (binding component properties, composite properties such as audit level settings and payload validation status, and policy attachments). ■ `runtime`: Postdeployment runtime changes are included (rules dictionary and domain value maps (DVMs)).
sarFile	Absolute path of a SAR file to generate.
compositeName	Name of the composite to export.
revision	Revision of the composite to export.

9.10.1.3 Examples

This example exports composite without including any postdeployment changes.

```
wls:/mydomain/ServerConfig> sca_exportCompositeMb('none', '/tmp/sca_HelloWorld_
rev1.0.jar', 'HelloWorld', '1.0')
```

This example exports a composite with all postdeployment updates.

```
wls:/mydomain/ServerConfig> sca_exportCompositeMb('all', '/tmp/sca_HelloWorld_
rev1.0-all.jar', 'HelloWorld','1.0')
```

This example exports a composite with property postdeployment updates.

```
wls:/mydomain/ServerConfig> sca_exportCompositeMb('property', '/tmp/sca_
HelloWorld_rev1.0-prop.jar', HelloWorld', '1.0')
```

This example exports a composite with runtime/metadata postdeployment updates.

```
wls:/mydomain/ServerConfig> sca_exportCompositeMb('runtime', '/tmp/sca_HelloWorld_
rev1.0-runtime.jar','HelloWorld', '1.0')
```

9.10.2 sca_exportUpdatesMb

Command Category: Application Export and Import Commands

Use with WLST: Online

9.10.2.1 Description

Exports postdeployment changes of a SOA composite application into a JAR file.

9.10.2.2 Syntax

```
sca_exportUpdatesMb(updateType, jarFile, compositeName, revision)
```

Argument	Definition
updateType	Type of postdeployment changes to be exported: `all`, `property`, or `runtime`.
jarFile	Absolute path of a JAR file to generate.
compositeName	Name of the composite to export.

Argument	Definition
revision	Revision of the composite to export.

9.10.2.3 Examples

The following example exports all postdeployment updates.

```
wls:/mydomain/ServerConfig>  sca_exportUpdatesMb('all',
'/tmp/all-HelloWorld_rev1.0.jar','HelloWorld', '1.0')
```

The following example exports property postdeployment updates.

```
wls:/mydomain/ServerConfig> sca_exportUpdatesMB('property',
'/tmp/prop-HelloWorld_ rev1.0.jar', 'HelloWorld', '1.0')
```

The following example exports runtime/metadata postdeployment updates.

```
wls:/mydomain/ServerConfig> sca_exportUpdatesMB('runtime',
'/tmp/runtime-HelloWorld_ rev1.0.jar', 'HelloWorld', '1.0')
```

9.10.3 sca_importUpdatesMb

Command Category: Application Export and Import Commands

Use with WLST: Online

9.10.3.1 Description

Imports postdeployment changes of a SOA composite application.

9.10.3.2 Syntax

```
sca_importUpdatesMb(jarFile, compositeName, revision)
```

Argument	Definition
jarFile	Absolute path of a JAR file that contains postdeployment changes.
compositeName	Name of the composite to which the postdeployment changes are imported.
revision	Revision of the composite to which the postdeployment changes are imported.

9.10.3.3 Examples

The following example imports postdeployment changes of a SOA composite application.

```
wls:/mydomain/ServerConfig> sca_importUpdatesMb('/tmp/all-HelloWorld_rev1.0.jar',
'HelloWorld', '1.0')
```

9.10.4 sca_exportSharedDataMb

Command Category: Application Export and Import Commands

Use with WLST: Online

9.10.4.1 Description

Exports shared data of a given pattern into a JAR file.

9.10.4.2 Syntax

```
sca_exportSharedDataMb(jarFile, pattern)
```

Argument	Definition
jarFile	Absolute path of a JAR file to generate.
pattern	The file pattern supported by MDS transfer APIs. Use the semicolon delimiter (;) if more than one pattern is specified. Exclude the shared data namespace /apps in the pattern. For example: /Project1/**;/Project2/** This example exports all documents under /apps/Project1 and /apps/Project2.

9.10.4.3 Examples

This example exports shared data of given pattern into a JAR file.

```
wls:/mydomain/ServerConfig> sca_exportSharedDataMb('/tmp/MySharedData.jar',
'/Project1/**')
```

9.11 SOA Composite Application Partition Management Commands

Use the deployment commands, listed in Table 9–11, to manage partitions. Partitioning enable you to logically group different revisions of your SOA composite applications into separate sections. This is similar to the concept of domains in the 10.1.*x* releases of Oracle BPEL Process Manager.

Table 9–11 SOA Composite Application Partition Management Commands for WLST Configuration

Use this command...	To...	Use with WLST...
sca_createPartition	Create a partition.	Online
sca_deletePartition	Undeploy all SOA composite applications in a partition before deleting the partition.	Online
sca_startCompositesInPartition	Start all SOA composite applications in a partition.	Online
sca_stopCompositesInPartition	Stop all SOA composite applications in a partition.	Online
sca_activateCompositesInPartition	Activate all SOA composite applications in a partition.	Online
sca_retireCompositesInPartition	Retire all SOA composite applications in a partition.	Online
sca_listPartitions	List all partitions in the SOA Infrastructure.	Online
sca_listCompositesInPartition	List all composites in a specific partition.	Online

9.11.1 sca_createPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

9.11.1.1 Description

Creates a partition.

9.11.1.2 Syntax

```
sca_createPartition(partitionName)
```

Argument	Definition
partitionName	The name of the partition.

9.11.1.3 Examples

This example creates a partition named `myPartition`.

```
wls:/mydomain/ServerConfig> sca_createPartition('myPartition')
```

9.11.2 sca_deletePartition

Command Category: Application Partition Management Commands

Use with WLST: Online

9.11.2.1 Description

Undeploys all composites in a partition before deleting the partition.

9.11.2.2 Syntax

```
sca_deletePartition(partitionName)
```

Argument	Definition
partitionName	The name of the partition.

9.11.2.3 Examples

This example undeploys all composites in the myPartition partition before deleting the partition.

```
wls:/mydomain/ServerConfig> sca_deletePartition('myPartition')
```

9.11.3 sca_startCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

9.11.3.1 Description

Starts all composites in a partition.

9.11.3.2 Syntax

```
sca_startCompositesInPartition(partitionName)
```

Argument	Definition
partitionName	The name of the partition.

9.11.3.3 Examples

This example starts all composites in the myPartition partition.

```
wls:/mydomain/ServerConfig> sca_startCompositesInPartition('myPartition')
```

9.11.4 sca_stopCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

9.11.4.1 Description

Stops all composites in a partition.

9.11.4.2 Syntax

```
sca_stopCompositesInPartition(partitionName)
```

Argument	Definition
partitionName	The name of the partition.

9.11.4.3 Examples

This example stops all composites in the myPartition partition.

```
wls:/mydomain/ServerConfig> sca_stopCompositesInPartition('myPartition')
```

9.11.5 sca_activateCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

9.11.5.1 Description

Activates all composites in a partition.

9.11.5.2 Syntax

```
sca_activateCompositesInPartition(partitionName)
```

Argument	Definition
partitionName	The name of the partition.

9.11.5.3 Examples

This example activates all composites in the myPartition partition.

```
wls:/mydomain/ServerConfig> sca_activateCompositesInPartition('myPartition')
```

9.11.6 sca_retireCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

9.11.6.1 Description

Retires all composites in a partition.

9.11.6.2 Syntax

```
sca_retireCompositesInPartition(partitionName)
```

Argument	Definition
partitionName	The name of the partition.

9.11.6.3 Examples

This example retires all composites in the myPartition partition.

```
wls:/mydomain/ServerConfig> sca_retireCompositesInPartition('myPartition')
```

9.11.7 sca_listPartitions

Command Category: Application Partition Management Commands

Use with WLST: Online

9.11.7.1 Description

Lists all partitions in the SOA Infrastructure.

9.11.7.2 Syntax

```
sca_listPartitions()
```

9.11.7.3 Examples

This example lists all partitions in the SOA Infrastructure.

```
wls:/mydomain/ServerConfig> sca_listPartitions()
```

9.11.8 sca_listCompositesInPartition

Command Category: Application Partition Management Commands

Use with WLST: Online

9.11.8.1 Description

Lists all composites in a partition.

9.11.8.2 Syntax

```
sca_listCompositesInPartition(partitionName)
```

Argument	Definition
partitionName	The name of the partition.

9.11.8.3 Examples

This example lists all composites in the `myPartition` partition.

```
sca_listCompositesInPartition(myPartition)
```

10

WebCenter Portal Custom WLST Commands

This chapter describes WebLogic Scripting Tool (WLST) commands for Oracle WebCenter Portal. These commands enable you to configure WebCenter Portal applications and components from the command-line. For additional details about WebCenter Portal 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 Portal WLST commands are only effective after you restart the Managed Server on which the WebCenter Portal application is deployed. The only exceptions are WLST commands for External Applications, Portlet Producers, and WebCenter Portal Import and Export.

WebCenter Portal WLST commands are described in the following sections:

10.1 WebCenter Portal WLST Command Categories

WebCenter Portal WLST commands are grouped into the following categories (Table 10–1).

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

Table 10–1 WLST Command Categories

Command Category	Description
General	Manage WebCenter Portal connections.
Analytics	Manage Analytics Collector connections and configure the Analytics Collector (on `WC_Utilities`).
Activity Graph	Manage Activity Graph metadata and provider configuration (on `WC_Utilities`).
Activity Stream	Archive and restore activity stream data generated for a WebCenter Portal application.
Content Repository	Manage content repository connections and configure the Documents service.
Discussions and Announcements	Manage discussions server connections and configure the Discussion and Announcement services.
External Applications	Manage external application connections.
Instant Messaging and Presence	Manage instant messaging and presence server connections and configure the Instant Messaging and Presence service.
Mail	Manage mail server connections and configure the Mail service.
Notifications	Manage settings for the Notifications service.
Personal Events	Manage personal event server connections.
Personalization	Manage personalization server connections.
Portlet Producers	Manage portlet producers.
RSS News Feeds	Manage proxy settings for the RSS service.
Search - Oracle SES Search	Manage Oracle Secure Enterprise Search (SES) connections and other search-related properties.

Table 10–1 (Cont.) WLST Command Categories

Command Category	Description
Search - Oracle SES Search Crawlers	Manage Oracle Secure Enterprise Search (SES) crawlers.
Search - WebCenter Portal Search	Manage search crawlers for the Spaces application.
Worklists	Manage BPEL server connections.
Spaces Application	Manage Spaces workflow settings and space metadata.
WebCenter Portal Identity Store	Configure options for searching a WebCenter Portal application's identity store.
WebCenter Portal Import and Export	Export and import Spaces applications, individual spaces and space templates, as well as producer metadata.

10.2 General

Use the General commands, listed in Table 10–2, to manage WebCenter Portal connections.

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

Table 10–2 General WLST Commands

Use This Command...	To...	Use with WLST...
deleteConnection	Delete any WebCenter Portal connection.	Online
setWebCenterServiceFrameworkConfig	Set WebCenter Portal Service Framework configuration properties.	Online
getWebCenterServiceFrameworkConfig	Return WebCenter Portal Framework configuration properties.	Online
webcenterErrorOccurred	Return status information for the last WebCenter Portal command executed.	Online
getWebCenterConnectionTypes	List all the WebCenter Portal connection types.	Online
cloneWebCenterManagedServer	Clone a WebCenter Portal Managed Server.	Online

10.2.1 deleteConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.2.1.1 Description

Deletes a named WebCenter Portal 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.

`deleteConnection` cannot be used to delete WebCenter Portal connections for the Personalization service. Instead, use deleteWCPSCMISConnection, deleteWCPSActivityGraphConnection, deleteWCPSPeopleConnection, or deleteWCPSCustomConnection.

10.2.1.2 Syntax

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

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.2.1.3 Example

The following example deletes a WebCenter Portal connection.

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

10.2.2 setWebCenterServiceFrameworkConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.2.2.1 Description

Sets WebCenter Portal Service Framework configuration properties, such as the Resource Action Handler class and display as popup properties.

10.2.2.2 Syntax

```
setWebCenterServiceFrameworkConfig(appName, [resourceActionHandlerClassName],
[resourceActionHandlerDisplayInPopup], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
resourceActionHandlerClassName	Optional. Class used by the Service Framework Resource Action Handler.

Argument	Definition
resourceActionHandlerDisplayInPopup	Optional. Indicates whether the Resource Action Handler displays resources in a popup or inline. Valid options are `1` (true) and `0` (false).
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.2.2.3 Example

The following example sets the WebCenter Portal Service Framework Resource Action Handler class to `my.company.ResourceActionHandler`:

```
wls:/wc_domain/domainRuntime>
setWebCenterServiceFrameworkConfig(appName='webcenter',
    resourceActionHandlerClassName='my.company.ResourceActionHandler')

Successfully set the WebCenter Portal service framework configuration.
Resource Action Handler class: my.company.ResourceActionHandler
To effect connection changes, you must restart the managed server on which the
WebCenter Portal application is deployed.
```

The following example sets only the WebCenter Portal Service Framework Resource Action Handler display as popup value to `1` (true):

```
wls:/wc_domain/domainRuntime>
setWebCenterServiceFrameworkConfig(appName='webcenter',
resourceActionHandlerDisplayInPopup=1)

Successfully set the WebCenter Portal service framework configuration.
Resource Action Handler Display In Popup: true
To effect connection changes, you must restart the managed server on which the
WebCenter Portal application is deployed.
```

10.2.3 getWebCenterServiceFrameworkConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.2.3.1 Description

Returns WebCenter Portal Service Framework configuration property settings, such as:

- `resourceActionHandlerClassName`: Class currently used by the WebCenter Portal Service Framework Resource Action Handler
- `resourceActionHandlerDisplayInPopup`: Indicates whether the Resource Action Handler displays resources in a popup or inline. Valid options are `1` (true) and `0` (false).

10.2.3.2 Syntax

```
getWebCenterServiceFrameworkConfig(appName, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.2.3.3 Example

The following example returns the service framework resource action handler class and display as popup properties, for the named application.

```
wls:/weblogic/serverConfig>getWebCenterServiceFrameworkConfig(appName='webcenter')
Resource Action Handler Class: my.company.ResourceActionHandler
Resource Action Handler Display In Popup: true
```

10.2.4 webcenterErrorOccurred

Module: Oracle WebCenter Portal

Use with WLST: Online

10.2.4.1 Description

Returns the status of last WebCenter Portal command executed.

Use the `webcenterErrorOccurred` command to determine the status of the last WebCenter Portal command executed. The command returns `1` if an error occurred or `0` otherwise.

10.2.4.2 Syntax

```
webcenterErrorOccurred ()
```

10.2.4.3 Example

The following example returns `1` if an error occurred:

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

10.2.5 getWebCenterConnectionTypes

Module: Oracle WebCenter Portal

Use with WLST: Online

10.2.5.1 Description

Lists all the WebCenter Portal connection types.

10.2.5.2 Syntax

```
getWebCenterConnectionTypes (appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.2.5.3 Example

The following example returns WebCenter Portal connection types for an application named `webcenter`:

```
wls:/mydomain/serverConfig> getWebCenterConnectionTypes(appName='webcenter')
```

10.2.6 cloneWebCenterManagedServer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.2.6.1 Description

Creates a new managed server with the same resources as a specified, base managed server.

10.2.6.2 Syntax

```
cloneWebCenterManagedServer(baseManagedServer, newManagedServer,
newManagedServerPort, [verbose])
```

Argument	Definition
baseManagedServer	Name of the base managed server.
newManagedServer	Name for the new, clone managed server.
newManagedServerPort	Port number for the new managed server.
verbose	Optional. Creates the managed server in verbose mode. Valid values are `1` and `0`. When set to `1`, additional progress information displays during the creation process which is useful for diagnostic purposes. The default is `0`.

10.2.6.3 Example

The following example creates a clone of the `WC_CustomPortal` managed server. The new managed server is named `WC_CustomPortal2`:

```
wls:/weblogic/serverConfig> cloneWebCenterManagedServer(baseManagedServer='WC_
CustomPortal', newManagedServer='WC_CustomPortal2', newManagedServerPort=1234)
```

10.3 Analytics

Analytics Collector Connections

Use the commands listed in Table 10–3 to manage Analytics Collector connections for a WebCenter Portal application. Events raised in WebCenter Portal applications using OpenUsage APIs can be sent to an Analytics Collector for use by Analytics and Activity Graph services.

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

Table 10–3 Analytics Collector Connection WLST Commands

Use this command...	To...	Use with WLST...
createAnalyticsCollectorConnection	Create a connection to an Analytics Collector for a WebCenter Portal application.	Online
setAnalyticsCollectorConnection	Edit an existing Analytics Collector connection.	Online
listAnalyticsCollectorConnections	List all of the Analytics Collector connections that are configured for a WebCenter Portal application.	Online
setDefaultAnalyticsCollectorConnection	Specify the default (or active) Analytics Collector connection for a WebCenter Portal application.	Online
listDefaultAnalyticsCollectorConnection	Return connection details for the Analytics Collector being used by a WebCenter Portal application.	Online

Analytics Collector and Cluster Configuration

Use the commands listed in Table 10–4 to configure event collection properties for the Analytics Collector that is deployed on `the WC_Utilities` managed server.

If you reconfigure the Analytics Collector or set up clustering, you must restart the managed server on which the Analytic Collector is deployed (`WC_Utilities`).

Table 10–4 Analytics Collector Configuration WLST Commands

Use this command...	To...	Use with WLST...
setAnalyticsCollectorConfig	Set Analytics Collector options, and cluster options if operating a clustered environment.	Online
listAnalyticsCollectorConfig	Return Analytics Collector settings.	Online
listAnalyticsEventTypes	List events currently registered with the Analytics Collector.	Online

10.3.1 createAnalyticsCollectorConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.3.1.1 Description

Creates a connection to an Analytics Collector for a named WebCenter Portal application.

Events raised in WebCenter Portal applications using OpenUsage APIs can be sent to an Analytics Collector for use by the Analytics and Activity Graph services.

While you can register multiple Analytics Collector connections for a WebCenter Portal application, only one Analytics Collector connection is used - the default (or active) connection where `default=1`.

10.3.1.2 Syntax

```
createAnalyticsCollectorConnection(appName, name, [isUnicast, collectorhost,
clusterName, collectorPort, isEnabled, timeout, default, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
name	Connection name. The name must be unique across all connection types within the WebCenter Portal application.
isUnicast	Optional. Specifies whether events are sent to a clustered Analytics Collector in multicast mode or whether a single Analytics Collector using unicast communication is required. Valid values are `1` (true) and `0` (false). The default value is `1` (unicast).
collectorHost	Optional. Host name where the Analytics Collector is running. The default value is `localhost`. Only required for unicast communication, that is, where `isUnicast='1'`.
clusterName	Optional. Name of the cluster where a clustered Analytics Collector is running. Only required for multicast communication, that is, where `isUnicast=0`.
collectorPort	Optional. Port on which the Analytics Collector listens for events. The default value is `31314`.
isEnabled	Optional. Specifies whether to send analytics events raised using OpenUsage APIs to the Analytics Collector. Valid values `1` (true) and `0` (false). The default value is `0`. Analytics events are sent to the Analytics Collector when `isEnabled=1` and `default=1`.
timeout	Optional. Length of time (in seconds) to wait for a response from the Analytics Collector. Default value is `30`. Only required for multicast communication, that is, where `isUnicast=0`.
default	Optional. Indicates whether this connection is the default (or active) Analytics Collector connection for the WebCenter Portal application. Valid values are `1` (true) and `0` (false). When set to `1`, the WebCenter Portal application sends events on this connection. When set to `0`, the connection is not used. The default for this argument is `0`. While you can register multiple Analytics Collector connections for a WebCenter Portal application, only one connection is used by Analytics and Activity Graph services—the default (or active) connection.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.3.1.3 Example

The following example creates a connection named `MyAnalyticsCollector` for a WebCenter Portal application named `webcenter`. Events are sent to a single Analytics Collector using *unicast* communication:

```
wls:/weblogic/serverConfig>createAnalyticsCollectorConnection(appName='webcenter',
connectionName='MyAnalyticsCollector', isUnicast=1,
collectorHost='myhost.com', collectorPort=31314, isEnabled=1, timeout=30,
default=1)
```

The following example creates a connection named `MyAnalyticsCollector` for a WebCenter Portal application named `webcenter`. Events are sent to a clustered Analytics Collector in *multicast* mode

```
wls:/weblogic/serverConfig>createAnalyticsCollectorConnection(appName='webcenter',
connectionName='MyAnalyticsCollector', isUnicast=0,
clusterName='collector-cluster',
ccollectorPort=31314, isEnabled=1, timeout=30, default=1)
```

10.3.2 setAnalyticsCollectorConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.3.2.1 Description

Edits an existing Analytics Collector connection for a named WebCenter Portal application.

Events raised in WebCenter Portal applications using OpenUsage APIs can be sent to an Analytics Collector for use by the Analytics and Activity Graph services.

While you can register multiple Analytics Collector connections for a WebCenter Portal application, only one Analytics Collector connection is used - the default (or active) connection.

10.3.2.2 Syntax

```
setAnalyticsCollectorConnection(appName, name, [isUnicast, collectorHost,
clusterName, collectorPort, isEnabled, timeout, default, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.

Argument	Definition
name	Connection name. The name must be unique (across all connection types within the WebCenter Portal application).
isUnicast	Optional. Specifies whether events are sent to a clustered Analytics Collector in multicast mode or whether a single Analytics Collector using unicast communication is required.
collectorHost	Optional. Host name where the Analytics Collector is running. The default value is `localhost`. Only required for unicast communication, that is, where `isUnicast=1`.
clusterName	Optional. Name of the cluster where a clustered Analytics Collector is running. Only required for multicast communication, that is, where `isUnicast=0`.
collectorPort	Optional. Port on which the Analytics Collector listens for events. The default value is `31314`.
isEnabled	Optional. Specifies whether to send analytics events raised using OpenUsage APIs to the Analytics Collector. Valid values `1` (true) and `0` (false). The default value is `false`. Analytics events are sent to the Analytics Collector when `isEnabled=1` and `default=1`.
timeout	Optional. Length of time (in seconds) to wait for a response from the Analytics Collector. Default value is 30. Only required for multicast communication, that is, where `isUnicast=0`.
default	Optional. Indicates whether this connection is the default (or active) Analytics Collector connection for the WebCenter Portal application. Valid values `1` (true) and `0` (false). When set to `1`, the WebCenter Portal application sends events on this connection. When set to `0`, the connection is not used. The default for this argument is `0`. While you can register multiple Analytics Collector connections for a WebCenter Portal application, only one connection is used by the Analytics and Activity Graph services— the default (or active) connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.3.2.3 Example

The following example updates host and port details for an existing Analytics Collector connection named `MyAnalyticsCollector`. On this connection, events are sent to a single Analytics Collector in *unicast* mode:

```
wls:/weblogic/serverConfig>setAnalyticsCollectorConnection(appName='webcenter',
connectionName='MyAnalyticsCollector', collectorHost='myhost.com',
collectorPort=31314)
```

The following example updates cluster, port, and timeout details for an existing Analytics Collector connection named `MyAnalyticsCollector`. On this connection, events are sent to a clustered Analytics Collector in *multicast* mode:

```
wls:/weblogic/serverConfig>setAnalyticsCollectorConnection(appName='webcenter',
connectionName='MyAnalyticsCollector', clusterName='collector-cluster',
collectorPort=31314, timeout=60)
```

10.3.3 listAnalyticsCollectorConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.3.3.1 Description

Lists connection names and details for all Analytics Collector connections that are configured for a named WebCenter Portal application.

10.3.3.2 Syntax

```
listAnalyticsCollectorConnections(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.3.3.3 Examples

The following example lists connection names and details for all the Analytics Collector connections that are currently configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig>listAnalyticsCollectorConnections(appName='webcenter')
```

10.3.4 setDefaultAnalyticsCollectorConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.3.4.1 Description

Specifies the default Analytics Collector connection for a named WebCenter Portal application.

The default Analytics Collector connection is used to send events raised in WebCenter Portal applications using OpenUsage APIs to an Analytics Collector for use by Analytics and Activity Graph services.

While you can register multiple Analytics Collector connections for a WebCenter Portal application, only one Analytics Collector connection is used - the default (or active) connection.

10.3.4.2 Syntax

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

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing Analytics Collector connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.3.4.3 Example

The following example configures the connection `MyAnalyticsCollector` for events raised in an application named `webcenter`:

```
wls:/weblogic/serverConfig> setDefaultAnalyticsCollectorConnection
(appName='webcenter', name='myAnalyticsCollector')
```

10.3.5 listDefaultAnalyticsCollectorConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.3.5.1 Description

Return details about the Analytics Collector connection that is currently configured for a WebCenter Portal application.

While you can register multiple Analytics Collector connections for a WebCenter Portal application, only one Analytics Collector connection is used—the default (or active) connection.

10.3.5.2 Syntax

```
listDefaultAnalyticsCollectorConnection(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.3.5.3 Examples

The following example returns details about the Analytics Collector connection that is currently configured for a WebCenter Portal application named `webcenter`:

```
wls:/weblogic/serverConfig>listDefaultAnalyticsCollectorConnection(appName='webcenter')
```

10.3.6 setAnalyticsCollectorConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.3.6.1 Description

Configure the Analytics Collector deployed on the `WC_Utilities` managed server. Additionally, in a clustered environment, use this commands to set cluster settings.

10.3.6.2 Syntax

```
setAnalyticsCollectorConfig(appName, [collectorHost, defaultPort, maxPort,
broadcastType, clusterEnabled, clusterName, clusterBroadcastFrequency, server,
applicationVersion])
```

Argument	Definition
appName	Name of the Analytics Collector application.
collectorHost	Optional. Name of the host on which the Analytics Collector is running. The default value is `localhost`.
defaultPort	Optional. Default port number on which the Analytics Collector listens. The default value is `31314`.
maxPort	Optional. Highest port number that the Analytics Collector can use when allocating a listener. This property is mostly used in a clustered environment where more than one collector is running in the same box. Each collector listens for incoming UDP messages on a free port within a given port range. The range is from the default port number to the maxPort number.
broadcastType	Optional. Indicates the network channel on which the Analytics Collector broadcasts a 'heartbeat' to advertise its location to event producers. Valid values are `Broadcast` and `Multicast`. ■ `Broadcast` - use the standard network broadcast channel. ■ `Multicast` - use a special fixed multicast address.
clusterEnabled	Optional. Indicates whether the Analytics Collector is deployed in a cluster. Valid values are `1` (`true`) and `0` (`false`). If set to `1`, `clusterName` must also be defined.
clusterName	Optional. Name of the Analytics Collector cluster. Only required when `clusterEnabled=1`

Argument	Definition
clusterBroadcastFrequency	Optional. Broadcast Analytics Collector listening information every 'n' seconds. The default frequency is 10 seconds. The Analytics Collector periodically broadcasts a 'heartbeat' to advertise its location (`hostName`). In a clustered environment, WebCenter Portal applications use the heartbeat to determine which Analytics Collectors are available.
server	Optional. Name of the managed server where the Analytics Collector is deployed. For example, `WC_Utilities`. 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 application is deployed.

10.3.6.3 Example

The following example changes the default port to `31315`:

```
wls:/weblogic/serverConfig>setAnalyticsCollectorConnection(appName='analytics-collector', defaultPort=31315)
```

10.3.7 listAnalyticsCollectorConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.3.7.1 Description

Returns Analytics Collector settings.

10.3.7.2 Syntax

```
listAnalyticsCollectorConfig(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Analytics Collector application.
server	Optional. Name of the managed server where the Analytics Collector is deployed. For example, `WC_Utilities`. 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 application is deployed.

10.3.7.3 Examples

The following command lists current settings for the Analytics Collector that is configured for an application named `webcenter`:

```
wls:/weblogic/serverConfig>listAnalyticsCollectorConfig(appName='analytics-collector')
```

This is sample output for an Analytics Collector in a clustered environment:

```
CollectorHost = localhost
CollectorDefaultPort = 31314
CollectorMaximumPort = 31318
```

```
BroadcastType = Multicast
ClusterEnabled = 1
ClusterName = myCluster
ClusterBroadcastFrequency = 55
```

This is sample output for a standalone Analytics Collector:

```
CollectorHost = localhost
CollectorDefaultPort = 31314
CollectorMaximumPort = 31314
BroadcastType = Multicast
ClusterEnabled =
ClusterName =
ClusterBroadcastFrequency = 55
```

10.3.8 listAnalyticsEventTypes

Module: Oracle WebCenter Portal

Use with WLST: Online

10.3.8.1 Description

Lists all the events currently registered with the Analytics Collector.

10.3.8.2 Syntax

```
listAnalyticsEventTypes(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Analytics Collector application.
server	Optional. Name of the managed server where the Analytics Collector is deployed. For example, `WC_Utilities`. 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 application is deployed.

10.3.8.3 Examples

The following command lists all the events currently registered with the Analytics Collector for use by a WebCenter Portal application named `webcenter`:

```
wls:/weblogic/serverConfig>listAnalyticsEventTypes(appName='webcenter')
```

Sample output:

```
{HTTP://WWW.ORACLE.COM/ANALYTICS/WC}DISCUSSION_ANNOUNCEMENTEDIT
{HTTP://WWW.ORACLE.COM/ANALYTICS/WC}DISCUSSION_TOPICDELETE
{HTTP://WWW.ORACLE.COM/ANALYTICS/WC}PAGEEDIT
{HTTP://WWW.ORACLE.COM/ANALYTICS/WC}DOCLIB_DOCUMENTCREATE
{HTTP://WWW.ORACLE.COM/ANALYTICS/WC}LOGINS
...
```

10.4 Activity Graph

Use the commands listed in Table 10–5 to manage Activity Graph system properties and metadata.

Configuration changes made using the setAGProperty WLST command are only effective after your restart the managed server on which the Activity Graph application is deployed (`WC_Utilities`). For all other commands, configuration changes are effective immediately.

See also, "Managing the Activity Graph Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.

Table 10–5 Activity Graph WLST Commands

Use this command...	To...	Use with WLST...
exportAGMetadata	Export Activity Graph metadata definitions to an XML file.	Online
importAGMetadata	Import Activity Graph metadata definitions from an XML file.	Online
exportAGProviderConfiguration	Export provider configuration, for a given provider, to an Activity Graph metadata definition file.	Online
deleteAllAGMetadata	Delete all the Activity Graph metadata that is defined for a WebCenter application.	Online
deleteAGAction	Delete the metadata for an action registered with Activity Graph.	Online
deleteAGNodeClass	Delete the metadata for a node class registered with Activity Graph.	Online
deleteAGSimilarityCalculation	Delete the metadata for a similarity calculation registered with Activity Graph.	Online
deleteAGRankCalculation	Delete the metadata for a rank calculation registered with Activity Graph.	Online
deleteAGProviderAssignment	Delete the metadata for a provider assignment registered with Activity Graph.	Online
deleteAGQRPPRegistration	Delete the metadata for a QRPP registered with Activity Graph.	Online
deleteAGProviderConfiguration	Delete the metadata for a provider configuration registered with Activity Graph.	Online
renameAGAction	Change the URN of an action registered with Activity Graph.	Online
renameAGNodeClass	Change the URN of a node class registered with Activity Graph.	Online
setAGProperty	Set a system property for Activity Graph.	Online
getAGProperty	Return the current setting for a given Activity Graph property.	Online
setAGPasswordCredential	Set credentials (user name and password) for an Activity Graph property.	Online

10.4.1 exportAGMetadata

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.1.1 Description

Exports Activity Graph metadata definitions to an XML file.

10.4.1.2 Syntax

```
exportAGMetadata(appName, directoryPath, definitionFileName,
includeProviderConfigurations,[server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
directoryPath	Destination directory for the XML file that will be generated. If you specify a directory that does not exist then it will be created.
definitionFileName	Name for the XML file that will be generated. If a file with the same name exists in the destination directory then it will be overwritten.
includeProviderConfigurations	Determines whether the export includes provider configuration metadata. Valid values are `1` (true) and `0` (false). Provider configurations are a subset of Activity Graph metadata that you may want to manage separately from the other metadata.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.1.3 Example

The following example exports Activity Graph metadata definitions to an XML file named `ag-metadata.xml`, at the specified location:

```
wls:/weblogic/serverConfig> exportAGMetadata(appName='activitygraph-engines',
directoryPath='/scratch/myAGmetadata', definitionFileName='ag-metadata.xml',
includeProviderConfigurations='1')
```

10.4.2 importAGMetadata

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.2.1 Description

Imports Activity Graph metadata definitions from an XML file.

On import, new Activity Graph metadata definitions are created on the target and existing definitions are overwritten.

10.4.2.2 Syntax

```
importAGMetadata(appName, definitionFilePath, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
definitionFilePath	Relative path to the XML file containing metadata definitions. For example, `metadata/import-metadata.xml`.

Argument	Definition
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.2.3 Example

The following example imports Activity Graph metadata definitions from a file name `import-metadata.xml`:

```
wls:/weblogic/serverConfig> importAGMetadata(appName='activitygraph-engines',
definitionFilePath='metadata/import-metadata.xml')
```

10.4.3 exportAGProviderConfiguration

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.3.1 Description

Exports provider configuration, for a given provider, to an Activity Graph metadata definition file.

10.4.3.2 Syntax

```
exportAGProviderConfiguration(appName, directoryPath, definitionFileName, urn,
[server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
directoryPath	Destination directory for the XML file that will be generated. If you specify a directory that does not exist, then it will be created.
definitionFilePath	Name for the XML file that will be generated. If a file with the same name exists in the destination directory then it will be overwritten. Example
urn	URN for the Activity Graph provider to export.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.3.3 Example

The following example exports configuration information for the Activity Graph provider `oracle.webcenter.activitygraph.analytics` to an XML file named `'ag-provider-config.xml`, at the specified location:

```
wls:/weblogic/serverConfig>
exportAGProviderConfiguration(appName='activitygraph-engines',
```

```
directoryPath='/scratch/myAGmetadata',
definitionFileName='ag-provider-config.xml',
urn='oracle.webcenter.activitygraph.analytics')
```

10.4.4 deleteAllAGMetadata

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.4.1 Description

Deletes all the Activity Graph metadata that is defined for a WebCenter Portal application. The delete operation is immediate and non-reversible.

You can use this command in conjunction with the WLST command importAGMetadata to completely re-install Activity Graph metadata.

Note: Any data in the relation store, similarity store, and rank store will be deleted the next time the Activity Graph engines run.

10.4.4.2 Syntax

```
deleteAllAGMetadata(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.4.3 Example

The following example deletes all existing Activity Graph metadata:

```
wls:/weblogic/serverConfig> deleteAllAGMetadata(appName='activitygraph-engines')
```

10.4.5 deleteAGAction

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.5.1 Description

Deletes the metadata for an action that is currently registered with Activity Graph. The delete operation is immediate and non-reversible.

Note: Any data in the relation store that is associated with the action will be deleted the next time the Activity Graph engines run.

10.4.5.2 Syntax

```
deleteAGAction(appName, urn, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
urn	URN for the Activity Graph action to delete.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.5.3 Example

The following example deletes Activity Graph metadata for the `connect` action:

```
wls:/weblogic/serverConfig> deleteAGAction(appName='activitygraph-engines',
urn='connect')
```

10.4.6 deleteAGNodeClass

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.6.1 Description

Deletes the metadata for a node class that is currently registered with Activity Graph. The delete operation is immediate and non-reversible.

> **Note:** Any data in the relation store that is associated with the node class will be deleted the next time the Activity Graph engines run.

10.4.6.2 Syntax

```
deleteAGNodeClass(appName, urn, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
urn	URN for the Activity Graph node class to delete.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.6.3 Example

The following example deletes Activity Graph metadata for the node class `WC.wiki-page` action:

```
wls:/weblogic/serverConfig> deleteAGNodeClass(appName='activitygraph-engines',
urn='WC.wiki-page')
```

10.4.7 deleteAGSimilarityCalculation

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.7.1 Description

Deletes the metadata for a similarity calculation that is currently registered with Activity Graph. The delete operation is immediate and non-reversible.

10.4.7.2 Syntax

```
deleteAGSimilarityCalculation(appName, urn, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
urn	URN for the Activity Graph similarity calculation to delete.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.7.3 Example

The following example deletes Activity Graph metadata for the similarity calculation `item-edit`:

```
wls:/weblogic/serverConfig>
deleteAGSimilarityCalculation(appName='activitygraph-engines', urn='item-edit')
```

10.4.8 deleteAGRankCalculation

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.8.1 Description

Deletes the metadata for a rank calculation that is currently registered with Activity Graph. The delete operation is immediate and non-reversible.

10.4.8.2 Syntax

```
deleteAGRankCalculation(appName, urn, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
urn	URN for the Activity Graph rank calculation to delete.

Argument	Definition
server	Optional. Name of the managed server where the application is deployed. For example, WC_Utilities. 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 Activity Graph application is deployed.

10.4.8.3 Example

The following example deletes Activity Graph metadata for the activity-rank calculation:

```
wls:/weblogic/serverConfig>
deleteAGRankCalculation(appName='activitygraph-engines', urn='activity-rank')
```

10.4.9 deleteAGProviderAssignment

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.9.1 Description

Deletes the metadata for a provider assignment that is currently registered with Activity Graph, that is, a provider assignment defined by the unique triple combination (action, sourceClass, trgClass).

The delete operation is immediate and non-reversible.

10.4.9.2 Syntax

```
deleteAGProviderAssignment(appName, actionURN, srcClasURN, trgClassURN [server,
applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always activitygraph-engines.
action*URN*	URN for the action.
srcClassURN	URN for the source node class.
trgClassURN	URN for the target node class.
server	Optional. Name of the managed server where the application is deployed. For example, WC_Utilities. 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 Activity Graph application is deployed.

10.4.9.3 Example

The following example deletes Activity Graph metadata for the provider assignment specified:

```
wls:/weblogic/serverConfig>
deleteAGRProviderAssignment(appName='activitygraph-engines', actionURN='connect',
srcClassURN='WC.user', trgClassURN='WC.user')
```

10.4.10 deleteAGQRPPRegistration

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.10.1 Description

Deletes the metadata for a QRPP (Query Result Post Processor) that is currently registered with Activity Graph.

The delete operation is immediate and non-reversible.

10.4.10.2 Syntax

```
deleteAGQRPPRegistration(appName, urn [server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
`urn`	URN for the QRPP to delete.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.10.3 Example

The following example deletes Activity Graph metadata for a QRPP named `Event store metadata QRPP`:

```
wls:/weblogic/serverConfig>
deleteAGQRPPRegistration(appName='activitygraph-engines', urn='Event store
metadata QRPP')
```

10.4.11 deleteAGProviderConfiguration

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.11.1 Description

Deletes the metadata for a provider configuration. The delete operation is immediate and non-reversible.

10.4.11.2 Syntax

```
deleteAGProviderConfiguration(appName, urn [server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
`urn`	URN for the Activity Graph provider to delete.

Argument	Definition
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.11.3 Example

The following example deletes configuration information for the Activity Graph provider `oracle.webcenter.activitygraph.analytics`:

```
wls:/weblogic/serverConfig>
deleteAGProviderConfiguration(appName='activitygraph-engines',
urn='oracle.webcenter.activitygraph.analytics')
```

10.4.12 renameAGAction

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.12.1 Description

Changes the URN of an action that is currently registered with Activity Graph. Any data in the relation store that is associated with the action is preserved.

> **Note:** This command does not delete the action and create an action with a different name as this causes data associated with the original action to be deleted.

10.4.12.2 Syntax

```
renameAGAction(appName, currentURN, newURN,[server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
`currentURN`	Current action URN.
`newURN`	New action URN.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.12.3 Example

The following example changes the `connect` action URN to `people-connect`:

```
wls:/weblogic/serverConfig> renameAGAction(appName='activitygraph-engines',
currentURN='connect', newURN='connect')
```

10.4.13 renameAGNodeClass

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.13.1 Description

Changes the URN of a node class that is currently registered with Activity Graph. Any data in the relation store that is associated with the node class is preserved.

> **Note:** This command does not delete the node class and create a node class with a different name as this would cause data associated with the original node class to be deleted.

10.4.13.2 Syntax

```
renameAGNodeClass(appName, currentURN, newURN,[server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
`currentURN`	Current node class URN.
`newURN`	New node class URN.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.13.3 Example

The following example changes the `WC.user` node class URN to `WC.people`:

```
wls:/weblogic/serverConfig> renameAGNodeClass(appName='activitygraph-engines',
currentURN='WC.user', newURN='WC.people')
```

10.4.14 setAGProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.14.1 Description

Sets a system property for Activity Graph. This command sets a value based on the property's datatype (String, Integer, Float, Boolean).

Activity Graph system properties include settings for:

- Oracle Secure Enterprise Search (SES) Admin API Web service connection (`oracle.webcenter.activitygraph.providers.datasources.ses.soap.admin.url` and `oracle.webcenter.activitygraph.providers.datasources.ses.soap.query.url`)

- Engine configuration (`oracle.webcenter.activitygraph.rankengine.enabled`)

See also, "Managing the Activity Graph Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter* for a list of system properties and their datatypes.

Configuration changes made using the `setAGProperty` WLST command are only effective after your restart the managed server on which the Activity Graph application is deployed (`WC_Utilities`).

10.4.14.2 Syntax

```
setAGProperty(appName, propertyName, propertyValue, propertyType,[server,
applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
`propertyName`	Name of the Activity Graph property.
`propertyValue`	Value for the Activity Graph property.
`propertyType`	Datatype of the property. Valid values are: `String`, `Int`, `Float` or `Boolean`. Values are case sensitive.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.14.3 Example

The following example enables the Rank Engine:

```
wls:/weblogic/serverConfig> setAGProperty(appName='activitygraph-engines',
propertyName='oracle.webcenter.activitygraph.rankengine.enabled',
propertyValue='true', propertyType='boolean')
```

10.4.15 getAGProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.15.1 Description

Returns the current setting for a given Activity Graph property.

See also, "Managing the Activity Graph Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter* for a list of valid system properties.

10.4.15.2 Syntax

```
getAGProperty(appName, propertyName, propertyType [server, applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
`propertyName`	Name of the Activity Graph property.
`propertyType`	Datatype of the property. Valid values are: `String`, `Int`, `Float` or `Boolean`. Values are case sensitive.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.15.3 Example

The following example returns the current value of the system property `oracle.webcenter.activitygraph.providers.datasources.ses.soap.admin.url`:

```
wls:/weblogic/serverConfig> getAGProperty(appName='activitygraph-engines',
propertyName='oracle.webcenter.activitygraph.providers.datasources.ses.soap.admin.
url', propertyType='String')
```

10.4.16 setAGPasswordCredential

Module: Oracle WebCenter Portal

Use with WLST: Online

10.4.16.1 Description

Sets credentials (user name and password) for an Activity Graph credential property.

See also, "Managing the Activity Graph Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter* for a list of properties with the `PasswordCredential` datatype, for example, `oracle.webcenter.activitygraph.providers.datasources.ses.soap.admin.credential`.

10.4.16.2 Syntax

```
setAGPasswordCredentialProperty(appName, propertyName, userName, password,[server,
applicationVersion])
```

Argument	Definition
appName	Name of the Activity Graph application in which to perform this operation—always `activitygraph-engines`.
`propertyName`	Name of the Activity Graph property that specifies credentials (and has `PasswordCredential` datatype).
`userName`	User name associated with the credential property.
`password`	Password associated with the user name specified.

Argument	Definition
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Utilities`. 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 Activity Graph application is deployed.

10.4.16.3 Example

The following example sets user name and password credentials for the Oracle SES Admin tool:

```
wls:/weblogic/serverConfig> setAGProperty(appName='activitygraph-engines',
propertyName='oracle.webcenter.activitygraph.providers.datasources.ses.soap.admin.
credential',
userName='myname', password='GuessWhat')
```

10.5 Activity Stream

Use the commands listed in Table 10–6 to archive and restore activity stream data generated for a WebCenter Portal application.

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

Table 10–6 Activity Stream WLST Commands

Use this command...	To...	Use with WLST...
archiveASByDate	Archive activity stream data that is older than a specified date.	Online
archiveASByDeletedObjects	Archive activity stream data associated with deleted objects.	Online
archiveASByClosedSpaces	Archive activity stream data associated with Spaces that are currently closed.	Online
archiveASByInactiveSpaces	Archive activity stream data associated with Spaces that have been inactive since a specified date.	Online
restoreASByDate	Restore archived activity stream data from a specified date into production tables.	Online
truncateASArchive	Truncates activity stream archive data.	Online

10.5.1 archiveASByDate

Module: Oracle WebCenter Portal

Use with WLST: Online

10.5.1.1 Description

Archives activity stream data that is older than a specified date.

This command moves data from production tables to archive tables. Exceptions include WC_ACTOR_DETAIL and WC_OBJECT_DETAIL—data in these tables is copied to archive tables rather than moved.

Rows in WC_OBJECT_DETAIL that are not used by any activity element are deleted.

10.5.1.2 Syntax

```
archiveASByDate(appName, year, month, day, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
year	Year before which to archive activity stream data. For example, 2009.
month	Month before which to archive activity stream data. For example, enter 1 for January, 2 for February, and so on.
day	Day of the month before which to archive activity stream data.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.5.1.3 Example

The following example archives activity stream data that is older than October 1, 2009:

```
wls:/weblogic/serverConfig> archiveASByDate(appName='webcenter', year=2009,
month=10, day=1)
```

10.5.2 archiveASByDeletedObjects

Module: Oracle WebCenter Portal

Use with WLST: Online

10.5.2.1 Description

Archives activity stream data associated with deleted objects.

This command moves data from production tables to archive tables, except for WC_ACTOR_DETAIL—data in this table is copied to the archive table rather than moved.

Rows in WC_OBJECT_DETAIL that satisfy the criteria (in this case, deleted objects) are deleted.

10.5.2.2 Syntax

```
archiveASByDeletedObjects(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.5.2.3 Example

The following example archives activity stream data associated with deleted objects:

```
wls:/weblogic/serverConfig> archiveASByDeletedObjects(appName='webcenter')
```

10.5.3 archiveASByClosedSpaces

Module: Oracle WebCenter Portal

Use with WLST: Online

10.5.3.1 Description

Archives activity stream data associated with Spaces that are currently closed.

This command moves data from production tables to archive tables, except for WC_ACTOR_DETAIL—data in this table is copied to the archive table rather than moved. Rows in WC_OBJECT_DETAIL that satisfy the criteria (in this case, objects involved in activities of Spaces that are closed) are deleted.

10.5.3.2 Syntax

```
archiveASByClosedSpaces(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.5.3.3 Example

The following example archives activity stream data associated with Spaces that are currently closed:

```
wls:/weblogic/serverConfig> archiveASByClosedSpaces(appName='webcenter')
```

10.5.4 archiveASByInactiveSpaces

Module: Oracle WebCenter Portal

Use with WLST: Online

10.5.4.1 Description

Archives activity stream data associated with spaces that have been inactive since a specified date. An inactive space is an open or closed space in which there has been no activity since the specified date.

This command moves data from production tables to archive tables, except for WC_ACTOR_DETAIL—data in this table is copied to the archive table rather than moved.

Rows in WC_OBJECT_DETAIL that satisfy the criteria (in this case, objects involved in activities of spaces that have been inactive since the specified date) are deleted.

10.5.4.2 Syntax

```
archiveASByInactiveSpaces(appName, year, month, day, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
year	Year the space became inactive. For example, 2009.
month	Month the space became inactive. For example, enter 1 for January, 2 for February, and so on.
day	Day of the month the space became inactive.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, WC_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 Spaces application is deployed.

10.5.4.3 Example

The following example archives activity stream data associated with spaces that have been inactive (no activities have occurred, regardless of open or closed status) since October 1, 2009:

```
wls:/weblogic/serverConfig> archiveASByInactiveSpaces(appName='webcenter',
year=2009, month=10, day=1)
```

10.5.5 restoreASByDate

Module: Oracle WebCenter Portal

Use with WLST: Online

10.5.5.1 Description

Restores archived activity stream data from a specified date into production tables.

This command moves data from archive tables to production tables, except for WC_ACTOR_DETAIL—data in this table is not restored because data is not deleted from this table during the archive process.

Rows that already exist in the production tables are not changed during the restore process.

10.5.5.2 Syntax

```
restoreASByDate(appName, year, month, day, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
year	Year from which to restore activity stream data. For example, 2009.
month	Month from which to restore activity stream data. For example, enter 1 for January, 2 for February, and so on.
day	Day of the month from which to restore activity stream data.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.5.5.3 Example

The following example restores activity stream data archived since October 1, 2009:

```
wls:/weblogic/serverConfig>restoreASByDate(appName='webcenter', year=2009,
month=10, day=1)
```

10.5.6 truncateASArchive

Module: Oracle WebCenter Portal

Use with WLST: Online

10.5.6.1 Description

Truncates activity stream archive data.

10.5.6.2 Syntax

```
truncateASArchive(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.5.6.3 Example

The following example truncates activity stream archive data:

```
wls:/weblogic/serverConfig>truncateASArchive(appName='webcenter')
```

10.6 Content Repository

Use the commands listed in Table 10–7 to manage content repository connections and configure the Documents service for a WebCenter Portal application.

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

Table 10–7 Content Repository WLST Commands

Use this command...	To...	Use with WLST...
createJCRContentServerConnection	Create a connection to an Oracle WebCenter Content repository.	Online
setJCRContentServerConnection	Edit an existing Oracle WebCenter Content repository connection.	Online
listJCRContentServerConnections	List individual or all Oracle WebCenter Content repository connections that are configured for a WebCenter Portal application.	Online
createJCRPortalConnection	Create an Oracle Portal repository connection.	Online
setJCRPortalConnection	Edit an existing Oracle Portal repository connection.	Online
listJCRPortalConnections	List all Oracle Portal connections that are configured for a WebCenter Portal 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 Portal application.	Online
createJCRSharePointConnection	Create a Microsoft SharePoint 2007 repository connection.	Online
setJCRSharePointConnection	Edit a Microsoft SharePoint 2007 repository connection.	Online
listJCRSharePointConnections	List all Microsoft SharePoint 2007 connections that are configured for a WebCenter Portal application.	Online
listDocumentsSpacesProperties	List properties for the back-end Content Server that is being used by the Spaces application.	Online
setDocumentsSpacesProperties	Modify properties for the back-end Content Server used by the Spaces application.	Online
deleteDocumentsSpacesProperties	Delete properties for the back-end Content Server used by the Spaces application.	Online

10.6.1 createJCRContentServerConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.1.1 Description

Creates a connection to an Oracle WebCenter Content repository for a named WebCenter Portal application.

10.6.1.2 Syntax

```
createJCRContentServerConnection(appName, name, socketType, [url, serverHost,
serverPort, keystoreLocation, keystorePassword, privateKeyAlias,
privateKeyPassword, webContextRoot, clientSecurityPolicy,
cacheInvalidationInterval, binaryCacheMaxEntrySize,
adminUsername, adminPassword, extAppId, timeout, isPrimary, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal application.
socketType	Specifies whether Oracle WebCenter Content's 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 Content Server. The client IP address must be added to the list of authorized addresses in the Content Server. In this case, the client is the machine on which Oracle WebCenter Portal is running. ■ **socketssl**—Use an `intradoc` socket connection to connect to the 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 the Spaces application). ■ **web**—Use an HTTP(S) connection to connect to the Content Server. Note that for the Spaces application, this option is not suitable for the active connection, that is, the back-end Content Server. repository that is being used to store space-specific documents and Home space documents, because it does not allow identity propagation. ■ **jaxws**—Use a Java API for XML Web Services connection to connect to the Content Server.
url	Optional. Content Server URL. Required only if `socketType` is set to `web` or `jaxws`. 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 Content Server is running. Required if `socketType` is set to `socket` or `socketssl`.
serverPort	Optional. Port on which the 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. This property corresponds to the `IntradocServerPort` setting in the Content Server configuration file, which defaults to port `4444`.

Argument	Definition
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
webContextRoot	Optional. Web server context root for the Content Server. Use the format `/<context_root>`. For example, `/cs`. When specified, several Oracle WebCenter Content features based on iFrame are available in the WebCenter Portal application. This includes: ■ **Associating a content profile with files when uploading new or updated files to Content Server.** For more information, see "Uploading New Files" and "Uploading a New Version of an Existing File" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*. ■ **Using the document review functionality available in Oracle AutoVue.** For more information, see "Reviewing and Collaborating on Documents Using AutoVue" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*. ■ **Editing advanced document properties.** For more information, see "Working with File Properties" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*. ■ **Viewing folder and file workflow details.** For more information, see "Viewing Workflow Information" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*. ■ **Previewing files in a slide viewer.** For more information, see "Opening a File" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*. ■ **Site Studio integration** For more information, see *Oracle Fusion Middleware User's Guide for Oracle WebCenter*. `webContextRoot` is only applicable when `IDENTITY_PROPAGATION` is used for authentication, that is, when `extAppId` is set to an empty string. **Note:** To fully enable these Oracle WebCenter Content features you must access the WebCenter Portal application through Oracle HTTPS Server (OHS) to expose Content Server and the WebCenter Portal application under the same host and port. Both the WebCenter Portal application and Content Server must also use single sign on. For information about setting up OHS to front-end WebCenter Portal applications, see "Content Server - Configuration" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*. If your WebCenter Portal application is connected to multiple Content Servers, Oracle recommends that each Content Server has a unique Web Server Context Root so that OHS re-direction works correctly.
clientSecurityPolicy	Optional. Client security policy to be used when the `socketType` is `jaxws`. For example: `oracle/wss11_saml_token_with_message_protection_service_policy`
cacheInvalidationInterval	Optional. Frequency between checks for external Content Server content changes (in minutes). WebCenter Portal automatically clears items that have changed from the cache. Defaults to 0 which means that cache invalidation is disabled. The *minimum* interval is 2 minutes.
binaryCacheMaxEntrySize	Optional. Maximum cacheable size (in bytes) for Content Server binary documents. Documents larger than this size are not cached by WebCenter Portal. Defaults is 102400 bytes (100K). Tune this value based on your machine's memory configuration and the types of binary documents that you expect to cache.

Argument	Definition
adminUsername	Optional. User name with administrative rights for this Content Server instance. This user will be used to fetch content type information based on profiles and track document changes for cache invalidation purpose. Defaults to `sysadmin`.
adminPassword	Optional. Password for the Content Server administrator specified in `adminUsername`. Required when `socketType` is set to `web`.
extAppId	Optional. External application used to authenticate users against the 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 Portal application and Content Server use the same identity store to authenticate users. Note that `extAppID` is mandatory when `socketType` is set to `web`.
timeout	Optional. Length of time allowed to log in to Content Server (in ms) before issuing a connection timeout message. If no timeout is set, there is no time limit for the login operation.
isPrimary	Optional. Valid string values are `1` (true) and `0` (false). `1` specifies that this connection is the primary connection used by the Documents service. This argument defaults to `0`. In the Spaces application, the primary connection is used to store space-specific content and Home space content.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.1.3 Examples

The following example creates a socket-based connection to an Oracle WebCenter Content repository 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=1)
```

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

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

The following example creates a JAX-WS (Java API for XML Web Services) connection to an Oracle WebCenter Content repository:

```
wls:/weblogic/serverConfig> createJCRContentServerConnection(appName='webcenter',
name='myContentServerConnection', socketType='jaxws',
url='http://myhost.com:9044/idcnativews', clientSecurityPolicy='oracle/wss10_saml_
token_client_policy')
```

10.6.2 setJCRContentServerConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.2.1 Description

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

10.6.2.2 Syntax

```
setJCRContentServerConnection(appName, name, [socketType, url, serverHost,
serverPort, keystoreLocation, keystorePassword, privateKeyAlias,
privateKeyPassword, webContextRoot, clientSecurityPolicy,
cacheInvalidationInterval, binaryCacheMaxEntrySize, adminUsername, adminPassword,
extAppId, timeout, isPrimary, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing Oracle WebCenter Content repository connection.
socketType	Optional. Specifies whether the Oracle WebCenter Content's 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 Content Server. The client IP address must be added to the list of authorized addresses in the Content Server. In this case, the client is the machine on which Oracle WebCenter Portal is running. ■ **socketssl**—Use an `intradoc` socket connection to connect to the 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 the Spaces application). ■ **web**—Use an HTTP(S) connection to connect to the Content Server. Note that for the Spaces application, this option is not suitable for the back-end Content Server repository that is being used to store space-specific documents and Home space documents, because it does not allow identity propagation. ■ **jaxws**—Use a Java API for XML Web Services connection to connect to the Content Server.
url	Optional. Content Server URL. Required only if `socketType` is set to `web` or `jaxws`. URL should be in the format: `http://<hostname>:<port>/<web root>/<plugin root>` For example, `http://mycontentserver/cms/idcplg`.

Argument	Definition
serverHost	Optional. Host name of the machine where the Content Server is running. Required if `socketType` is set to `socket` or `socketssl`.
serverPort	Optional. Port on which the 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`.
webContextRoot	Optional. Web server context root for the Content Server. Use the format `/<context_root>`. For example, `/cs`. When specified, several Oracle WebCenter Content features based on iFrame, such as previewing files in a slide viewer, are available in the WebCenter Portal application. **Note:** To fully enable these features you must access the WebCenter Portal application through Oracle HTTPS Server (OHS) to expose Content Server and the WebCenter Portal application under the same host and port. In addition, both the WebCenter Portal application and the Content Server must use single sign on. For information about setting up OHS to front-end WebCenter Portal applications, see "Content Server - Configuration" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*. `webContextRoot` is only applicable when `IDENTITY_PROPAGATION` is used for authentication, that is, when `extAppId` is set to an empty string.
clientSecurityPolicy	Optional. Client security policy to be used when the `socketType` is `jaxws`. For example: `oracle/wss11_saml_token_with_message_protection_service_policy`
cacheInvalidationInterval	Optional. Frequency between checks for external Content Server content changes (in minutes). WebCenter Portal automatically clears items that have changed from the cache. Defaults to 0 which means that cache invalidation is disabled. The *minimum* interval is 2 minutes.
binaryCacheMaxEntrySize	Optional. Maximum cacheable size (in bytes) for Content Server binary documents. Documents larger than this size are not cached by WebCenter Portal. Defaults is 102400 bytes (100K). Tune this value based on your machine's memory configuration and the types of binary documents that you expect to cache.
adminUsername	Optional. User name with administrative rights for this Content Server instance. This user will be used to fetch content type information based on profiles and track document changes for cache invalidation purpose. Defaults to `sysadmin`.

Argument	Definition
adminPassword	Optional. Password for the Content Server administrator specified in `adminUsername`. Required when `socketType` is set to `web`.
extAppId	Optional. External application used to authenticate users against the 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 Portal application and Content Server use the same identity store to authenticate users.
timeout	Optional. Length of time allowed to log in to Content Server (in ms) before issuing a connection timeout message. If no timeout is set, there is no time limit for the login operation.
isPrimary	Optional. Valid string values are `1` (true) and `0` (false). `1` specifies that this connection is the primary connection used by the Documents service. This argument defaults to `0`. In the Spaces application, the primary connection is used to store space-specific content and Home space content.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.2.3 Examples

The following example edits a socket-based connection to an Oracle WebCenter Content repository.

```
wls:/weblogic/serverConfig>setJCRContentServerConnection(appName='webcenter',
name='myContentServerConnection', socketType='socket',
serverHost='myhost.com',  serverPort='4444',
extAppId='myExtApp', isPrimary=1)
```

The following example edits an SSL socket-based connection to an Oracle WebCenter Content repository.

```
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=1)
```

The following example edits a JAX-WS (Java API for XML Web Services) connection to an Oracle WebCenter Content repository:

```
wls:/weblogic/serverConfig> setJCRContentServerConnection(appName='webcenter',
socketType='jaxws', url='http://myhost.com:9044/idcnativews',
clientSecurityPolicy='oracle/wss10_saml_token_client_policy')
```

10.6.3 listJCRContentServerConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.3.1 Description

Without any arguments, this command lists all of the Oracle WebCenter Content repository connections that are configured for a named WebCenter Portal application.

10.6.3.2 Syntax

```
listJCRContentServerConnections(appName, [verbose],
[name, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Displays content repository connection details in verbose mode. Valid options are `1` (true) and `0`(false). When set to `1`, `listJCRContentServerConnections` lists all Oracle WebCenter Content repository connections that are configured for a WebCenter Portal application, along with their details. When set to `0`, only connection names are listed. This argument defaults to `0`.
name	Optional. Name of an existing Oracle WebCenter Content repository connection. When specified you can view connection details for a specific Oracle WebCenter Content repository 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.3.3 Examples

The following example lists Oracle WebCenter Content repository connections configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig> listJCRContentServerConnections(appName='webcenter')
```

The following example lists all properties of the Oracle WebCenter Content repository connection named `myContentServerConnection1`. The connection named `myContentServerConnection1` must exist and be an Oracle WebCenter Content repository 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=1, name='myContentServerConnection1')
```

10.6.4 createJCRPortalConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.4.1 Description

Creates an Oracle Portal repository connection.

10.6.4.2 Syntax

```
createJCRPortalConnection(appName, name, dataSource, [extAppId, isPrimary,
timeout, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal application.
dataSource	JNDI DataSource location used to connect to the portal. For example: `jdbc/MyPortalDS` The datasource must be on the server where the WebCenter Portal application is deployed.
extAppId	Optional. External application used to authenticate 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 Portal application and Oracle Portal use the same identity store to authenticate users.
timeout	Optional. Length of time allowed to log in to Oracle Portal (in ms) before issuing a connection timeout message. If no timeout is set, there is no time limit for the login operation.
isPrimary	Optional. Valid string values are `1` (true) and `0` (false). `1` specifies that this connection is the primary connection used by the Documents service. This argument defaults to `0`. In the Spaces application, the primary connection must be an Oracle WebCenter Content repository connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.4.3 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=1)
```

10.6.5 setJCRPortalConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.5.1 Description

Edits an existing Oracle Portal connection. This command requires that you specify values for either the `dataSource` or `isPrimary` argument.

10.6.5.2 Syntax

```
setJCRPortalConnection(appName, name, [dataSource, extAppId, timeout, isPrimary,
server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing Oracle Portal connection.
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 Portal application is deployed.
extAppId	Optional. External application used to authenticate 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 Portal application and Oracle Portal use the same identity store to authenticate users.
timeout	Optional. Length of time allowed to log in to Oracle Portal (in ms) before issuing a connection timeout message. If no timeout is set, there is no time limit for the login operation.
isPrimary	Optional. Valid string values are `1` (true) and `0` (false). `1` specifies that this connection is the primary connection used by the Documents service. When set to `0`, 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 the Spaces application, the primary connection must be an Oracle WebCenter Content repository connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.5.3 Example

The following example edits Oracle Portal repository connection details.

```
wls:/weblogic/serverConfig> setJCRPortalConnection(appName='webcenter',
name='myPortalConnection', dataSource='/newPortalDS', extAppId='myExtApp',
isPrimary=0)
```

10.6.6 listJCRPortalConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.6.1 Description

Without any arguments, this command lists all of the Oracle Portal connections that are configured for a named WebCenter Portal application.

10.6.6.2 Syntax

```
listJCRPortalConnections(appName, [verbose, name, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Displays content repository connection details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listJCRPortalConnections` lists all Oracle Portal connections that are configured for a WebCenter Portal application, along with their details. When set to `0`, only connection names are listed. This argument defaults to `0`.
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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.6.3 Example

The following example lists all of the Oracle Portal connections that are configured for a WebCenter Portal application.

```
wls:/weblogic/serverConfig> listJCRPortalConnections(appName='webcenter',
verbose=1, name='myPortalConnection')
```

10.6.7 createJCRFileSystemConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.7.1 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.

10.6.7.2 Syntax

```
createJCRFileSystemConnection(appName, name, path, [isPrimary, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal 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.
isPrimary	Optional. Valid values are `1` (true) and `0` (false). `1` specifies that this connection is the primary connection used by the Documents service. When set to `0`, 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 the Spaces application, the primary connection must be an Oracle WebCenter Content repository connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.7.3 Example

The following example creates a connection to a file system repository.

```
wls:/weblogic/serverConfig> createJCRFileSystemConnection(appName='webcenter',
name='FSAConnection', path='C:/ProjectDocuments')
```

10.6.8 setJCRFileSystemConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.8.1 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.

10.6.8.2 Syntax

```
setJCRFileSystemConnection(appName, name, [path, isPrimary, server,
applicationVersion])
```

Argument	Definition
appName	Application name in which 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.
isPrimary	Optional. Valid values are `1` (true) and `0` (false). When set to `1`, specifies that this connection is the primary connection used by the Documents service. When set to `0`, 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 the Spaces application, the primary connection must be an Oracle WebCenter Content repository connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.8.3 Example

The following example edits connection details for a file system repository.

```
wls:/weblogic/serverConfig> setJCRFileSystemConnection(appName='webcenter',
name='FSAConnection', path='C:/ProjectDocuments')
```

10.6.9 listJCRFileSystemConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.9.1 Description

Without any arguments, this command lists all of the file system connections that are configured for a named WebCenter Portal application.

> **Note:** File system connections *must not* be used in production or enterprise application deployments. This feature is provided for development purposes only.

10.6.9.2 Syntax

```
listJCRFileSystemConnections(appName, [verbose, name, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

Argument	Definition
verbose	Optional. Displays content repository connection details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listJCRFileSystemConnections` lists all file system connections that are configured for a WebCenter Portal application, along with their details. When set to `0`, only connection names are listed. This argument defaults to `0`.
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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.9.3 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=1)
```

10.6.10 createJCRSharePointConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.10.1 Description

Creates a connection to a Microsoft SharePoint 2007 repository.

10.6.10.2 Syntax

```
createJCRSharePointConnection(appName, name, url, [likeLimit, extAppId, timeout,
isPrimary, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal application.
url	Web address of the SharePoint site to which you want to connect. For example, if the SharePoint site address is `http://mysharepoint.mycompany.com`, enter this value for the `url` argument.

Argument	Definition
likeLimit	Optional. Number of characters the `LIKE` operator matches. The default is `64`. The SharePoint query language can use a `LIKE` keyword to constrain URL queries (document paths) that match a search pattern. By default, the `LIKE` operator supports a pattern match on strings up to 64 characters. Use this argument to specify a different character limit (any positive integer between 1 and 64) or enter `likeLimit=0` to disable the LIKE limit, that is, always send the full query string to the Microsoft SharePoint server. As Oracle recommends the default value (64), there is no need to specify this argument when you create a connection using the WLST command `createJCRSharePointConnection`. **Note:** Only specify a value above 64 if your SharePoint instance supports LIKE queries on URLs greater than 64 characters.
extAppId	Optional. External application used to authenticate users against the SharePoint repository. This value should match the name of an existing external application connection. See also listExtAppConnections. If `extAppId` is not set, the SharePoint repository connection will not work. `extAppId` can be set or changed at any time using the setJCRSharePointConnection command.
timeout	Optional. Length of time allowed to log in to the SharePoint repository (in ms) before issuing a connection timeout message. If no timeout is set, there is no time limit for the login operation.
isPrimary	Optional. Valid values are `1` (true) and `0` (false). `1` specifies that this connection is the primary connection used by the Documents service. The argument defaults to `0`. If this parameter is omitted, the primary connection used by the Documents service does not change. In the Spaces application, the primary connection *must* be an Oracle WebCenter Content repository connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.10.3 Example

The following example creates a connection to a Microsoft SharePoint site.

```
wls:/weblogic/serverConfig> createJCRSharePointConnection(appName='webcenter',
name='MySPConnection', url='http://mysharepoint.mycompany.com',
extAppId='myExtApp')
```

10.6.11 setJCRSharePointConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.11.1 Description

Edits an existing Microsoft SharePoint 2007 repository connection. This command requires that you specify values for `appName` and `name`, plus at least one additional argument.

10.6.11.2 Syntax

```
setJCRSharePointConnection(appName, name, [url, likeLimit, extAppId, timeout,
isPrimary, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing SharePoint connection.
url	Optional. Web address of the SharePoint site to which you want to connect. For example, if the SharePoint site address is `http://mysharepoint.mycompany.com`, enter this value for the `url` argument.
likeLimit	Optional. Number of characters the `LIKE` operator matches. The default is `64`. The SharePoint query language can use a `LIKE` keyword to constrain URL queries (document paths) that match a search pattern. By default, the `LIKE` operator supports a pattern match on strings up to 64 characters. Use this argument to specify a different character limit (any positive integer between 1 and 64) or enter `likeLimit=0` to disable the LIKE limit, that is, always send the full query string to the Microsoft SharePoint server. Oracle recommends the default value (64). The default is suitable in most instances so, typically, there is no need to set a new value. To reset the default, specify `likeLimit=''` or `likeLimit=64`. **Note:** Only specify a value above 64 if your SharePoint instance supports LIKE queries on URLs greater than 64 characters.
extAppId	Optional. External application used to authenticate users against the SharePoint repository. 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 current external application ID. If no external application is set, the SharePoint connection will not work.
timeout	Optional. Length of time allowed to log in to the SharePoint repository (in ms) before issuing a connection timeout message. If no timeout is set, there is no time limit for the login operation.
isPrimary	Optional. Valid values are `1` (true) and `0` (false). `1` specifies that this connection is the primary connection used by the Documents service. When set to `0`, 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 the Spaces application, the primary connection *must* be an Oracle WebCenter Content repository connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.11.3 Example

The following example edits SharePoint repository connection details.

```
wls:/weblogic/serverConfig> setJCRSharePointConnection(appName='webcenter',
name='MySPConnection', url='http://mysharepoint.mycompany.com',
extAppId='myExtApp')
```

10.6.12 listJCRSharePointConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.12.1 Description

Without any arguments, this command lists all of the SharePoint connections that are configured for a named WebCenter Portal application.

10.6.12.2 Syntax

```
listJCRSharePointConnections(appName, [verbose, name, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Displays SharePoint connection details in verbose mode. Valid options are `1` (`true`) and `0` (`false`). When set to `1`, `listJCRSharePointConnections` lists all SharePoint connections that are configured for a WebCenter Portal application, along with their details. When set to `0`, only connection names are listed. This argument defaults to `0`.
name	Optional. Name of an existing SharePoint connection. When specified you can view connection details for a specific SharePoint 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.12.3 Example

The following example lists the names of all the SharePoint connections that are configured for an application named `webcenter`.

```
wls:/weblogic/serverConfig> listJCRSharePointConnections(appName='webcenter')
```

The following example lists connection details for all of the SharePoint connections that are configured.

```
wls:/weblogic/serverConfig> listJCRSharePointConnections(appName='webcenter',
verbose=1)
```

10.6.13 listDocumentsSpacesProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.13.1 Description

Lists properties for the back-end Oracle WebCenter Content repository that is being used by the Spaces application to store space-specific documents and Home space documents. This command is only valid for the Spaces application.

10.6.13.2 Syntax

```
listDocumentsSpacesProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which to perform this operation—always `webcenter`.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.6.13.3 Example

The following example lists properties for the back-end Oracle WebCenter Content repository that is being used by a Spaces application (named `webcenter`) to store space-specific documents and Home space documents.

```
wls:/weblogic/serverConfig> listDocumentsSpacesProperties(appName='webcenter')

The Documents Spaces container is "/WebCenter1109"
The Documents repository administrator is "sysadmin"
The Documents Spaces container is "/WebCenter1109"
The Documents primary connection is "myOCSConnection"
```

10.6.14 setDocumentsSpacesProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.14.1 Description

Modifies properties for the back-end Oracle WebCenter Content repository that is being used by the Spaces application to store Space-related data. This command is only valid for the Spaces application.

10.6.14.2 Syntax

```
setDocumentsSpacesProperties(appName, [spacesRoot, adminUserName,
applicationName, server, applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which to perform this operation—always `webcenter`.
spacesRoot	Optional. Root folder under which the Spaces application 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`. This user will be used to create and maintain folders for Spaces application content and manage content access rights. Administrative privileges are required for this connection so that operations can be performed on behalf of Spaces users. Note that if you provide a value for this argument, you must also provide values for the `spacesRoot` and `applicationName` arguments.

applicationName	Optional. Unique Spaces application identifier. This name is used to separate data when multiple 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.
	The name specified here is also used to name document-related workflows, as follows:
	`<applicationName><WorkflowName>`
	`<applicationName><WorkflowStepName>`
	When naming workflows, only the first 14 characters of the Application Name are used.

server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.6.14.3 Examples

The following example modifies connection properties for the back-end Oracle WebCenter Content repository that is being used by the Spaces application to store space-specific documents and Home 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 WebCenter Content repository that is being used by the Spaces application to store space-specific documents and Home space documents.

```
wls:/weblogic/serverConfig> setDocumentsSpacesProperties(appName='webcenter',
adminUserName='sysadmin')
```

10.6.15 deleteDocumentsSpacesProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.6.15.1 Description

Deletes properties for the back-end Oracle WebCenter Content repository used by the Spaces application, that is the `adminUserName`, `applicationName`, and `spacesRoot`. This command is only valid for the Spaces application.

10.6.15.2 Syntax

```
deleteDocumentsSpacesProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which to perform this operation—always `webcenter`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WC_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 Portal application is deployed.

10.6.15.3 Example

The following example deletes connection properties (`adminUserName`, `applicationName`, `spacesRoot`) of the back-end Oracle WebCenter Content repository that is being used by the Spaces application.

```
wls:/weblogic/serverConfig> deleteDocumentsSpacesProperties(appName='webcenter')
```

10.7 Discussions and Announcements

Use the commands listed in Table 10–8 to manage discussions server connections for WebCenter Portal applications.

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

Table 10–8 Discussion and Announcement WLST Commands

Use this command...	To...	Use with WLST...
createDiscussionForumConnection	Create a new discussions server connection for a WebCenter Portal application.	Online
setDiscussionForumConnection	Edit an existing discussions server connection.	Online
setDefaultDiscussionForumConnection	Specify the default connection for the Discussions and Announcements services.	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
setDiscussionForumConnectionProperty	Set an additional discussions server connection property.	Online
deleteDiscussionForumConnectionProperty	Delete a discussions server connection property.	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
addDiscussionsServerAdmin	Grant system administrator permissions on the discussions server to a user or a group.	Online
syncDiscussionServerPermissions	Synchronizes discussion server permissions for subspaces that inherit security from their parent.	Online
setDiscussionsServerProperty	Set discussions server properties.	Online
getDiscussionsServerProperty	Return discussions server property values.	Online
removeDiscussionsServerProperty	Remove current discussions server property values.	Online

10.7.1 createDiscussionForumConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.1.1 Description

Creates a new discussions server connection for a named WebCenter Portal 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 Portal application, only one connection is used for discussion and announcement services - the default (or active) connection.

10.7.1.2 Syntax

```
createDiscussionForumConnection(appName, name, url, adminUser,
[timeout, default, policyURIForAuthAccess, policyURIForPublicAccess,
recipientKeyAlias])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal 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 Portal users. This account is mostly used for managing discussions and announcements in the Spaces application. 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 the Spaces application, that is, the category (on the discussions server) under which all Space-related discussions and announcements are stored.

Argument	Definition
policyURIForAuthAccess	Optional. URI to the SAML token based policy required for authenticated access to the discussions server Web service. The client policy specified must be compatible with the service policy that is configured for the `OWCDiscussionsServiceAuthenticated` endpoint in the discussions server. Out-of-the-box, the default service policy is WSS 1.0 SAML Token Service Policy (`oracle/wss10_saml_token_service_policy`). Valid client policy values include: ▪ `oracle/wss10_saml_token_client_policy` (WSS 1.0 SAML Token Client Policy) ▪ `oracle/wss11_saml_token_with_message_protection_client_policy` (WSS 1.1 SAML Token with Message Protection Client Policy) ▪ `GPA` (Global Policy Attachment) - Use GPA if your environment supports Global Policy Attachments. In addition, ensure that the default policy is detached from the `OWCDiscussionsServiceAuthenticated` endpoint in the discussions server using the WLST command detachWebServicePolicy or Enterprise Manager. See also "Managing the Announcements and Discussions Services" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
policyURIForPublicAccess	Optional. URI to the policy required to enforce message security and integrity for public access to the discussions server Web service. Default value is `oracle/no_authentication_client_policy`. The client policy specified must be compatible with the service policy that is configured for the `OWCDiscussionsServicePublic` endpoint in the discussions server. Out-of-the-box, a service policy is not configured for public access (`oracle/no_authentication_client_policy`). Valid client policy values include: ▪ `oracle/no_authentication_client_policy` (None) ▪ `oracle/wss11_with_message_protection_client_policy` (WSS 1.1 Message Protection Client Policy) ▪ `GPA` (Global Policy Attachment) - Use GPA if your environment supports Global Policy Attachments. In addition, you must ensure that the default policy attached to the `OWCDiscussionsServicePublic` endpoint in the discussions server is set to `oracle/no_authentication_service_policy`.
recipientKeyAlias	Optional. Recipient key alias to be used for message protected policies (applicable to the `OWCDiscussionsServicePublic` and `OWCDiscussionsServiceAuthenticated` endpoints). This is the alias to the certificate that contains the public key of the discussions server in the configured keystore. The default is null. See also "Configuring WS-Security for WebCenter Portal Applications and Components" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
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.

Argument	Definition
default	Optional. Indicates that this connection is the default connection for the Discussions and Announcements services. Valid options are `1` (true) and `0` (false). When set to `1`, the Discussions service and the Announcements service both use this connection. When set to `0`, the connection is not used. The default is `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.1.3 Example

The following example creates a discussions server connection for a WebCenter Portal application.

```
wls:/weblogic/serverConfig> createDiscussionForumConnection(appName='webcenter',
name='MyDiscussionServer', url='http://myhost.com:8888/owc_discussions',
adminUser='admin', policyURIForAuthAccess='oracle/wss10_saml_token_client_policy',
default=0)
```

10.7.2 setDiscussionForumConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.2.1 Description

Edits an existing discussions server connection. Use this command to update connection attributes.

The connection is created using the createDiscussionForumConnection command.

10.7.2.2 Syntax

```
setDiscussionForumConnection(appName, name, [url, adminUser,
policyURIForAuthAccess, policyURIForPublicAccess, recipientKeyAlias, timeout,
default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing discussions server connection.
url	Optional. URL to the discussions server.

Argument	Definition
adminUser	Optional. Name of the discussions server administrator. This account is used by the Discussions service to perform administrative operations on behalf of WebCenter Portal users. This account is mostly used for managing discussions and announcements in the Spaces application. 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 the Spaces application, that is, the category (on the discussions server) under which all Spaces discussion forums are stored.
policyURIForAuthAccess	Optional. URI to the SAML token based policy required for authenticated access to the discussions server Web service. The client policy specified must be compatible with the service policy that is configured for the `OWCDiscussionsServiceAuthenticated` endpoint in the discussions server. Out-of-the-box, the default service policy is WSS 1.0 SAML Token Service Policy (`oracle/wss10_saml_token_service_policy`). Valid client policy values include: ■ `oracle/wss10_saml_token_client_policy` (WSS 1.0 SAML Token Client Policy) ■ `oracle/wss11_saml_token_with_message_protection_client_policy` (WSS 1.1 SAML Token with Message Protection Client Policy) ■ `GPA` (Global Policy Attachment) - Use GPA if your environment supports Global Policy Attachments. In addition, ensure that the default policy is detached from the `OWCDiscussionsServiceAuthenticated` endpoint in the discussions server using the WLST command detachWebServicePolicy or Enterprise Manager. See also "Managing the Announcements and Discussions Services" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
policyURIForPublicAccess	Optional. URI to the policy required to enforce message security and integrity for public access to the discussions server Web service. Default value is `oracle/no_authentication_client_policy`. The client policy specified must be compatible with the service policy that is configured for the `OWCDiscussionsServicePublic` endpoint in the discussions server. Out-of-the-box, a service policy is not configured for public access (`oracle/no_authentication_client_policy`). Valid client values include: ■ `oracle/no_authentication_client_policy` (None) ■ `oracle/wss11_with_message_protection_client_policy` (WSS 1.1 Message Protection Client Policy) ■ `GPA` (Global Policy Attachment) - Use GPA if your environment supports Global Policy Attachments. In addition, you must ensure that the default policy attached to the `OWCDiscussionsServicePublic` endpoint in the discussions server is set to `oracle/no_authentication_service_policy`.

Argument	Definition
recipientKeyAlias	Optional. Recipient key alias to be used for message protected policy authentication. Only required when the discussion server connection is using a message protection-based security policy for authentication. The default is null. See also "Configuring WS-Security for WebCenter Portal Applications and Components" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
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 `1` (true) and `0` (false). When set to `1`, the Discussion and Announcement services use this connection. When set to `0`, the connection is not used. The default is `0`. To specify that the Discussion and Announcements service use this connection, change the value from `0` to `1`. 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 Portal 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.2.3 Example

The following example updates attributes for a secure discussions server connection named `MyDiscussionsServer`.

```
wls:/weblogic/serverConfig> setDiscussionForumConnection(appName='webcenter',
name='MyDiscussionServer', url='http://myhost.com:7786/owc_discussions',
adminUser='admin', policyURIForAuthAccess='oracle/wss10_saml_token_client_policy',
default=1)
```

10.7.3 setDiscussionForumConnectionProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.3.1 Description

Sets a discussions server connection property. Use this command when additional parameters are required to connect to your discussions server.

This commands provides 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 10–9, " Additional Discussion Connection Properties".

Table 10–9 Additional Discussion Connection Properties

Additional Connection Property	Description
application.root.category.id	(Spaces application only) Application root category ID on the discussions server under which all discussion forums are stored. For example, if set to 3, then all forums are stored inside the category 3.

10.7.3.2 Syntax

```
setDiscussionForumConnectionProperty(appName, name, key, value, [secure, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 `1` (true) and `0` (false). When `1`, the value is encrypted. The default option is `0`. Set to `1` if you are storing passwords.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.3.3 Example

The following example configures the location of the keystore certificate for a discussions server connection named `MyDiscussionServer`.

```
wls:/weblogic/serverConfig> setDiscussionForumConnectionProperty
(appName='webcenter', name='MyDiscussionServer',
key='application.root.category.id', value='3')
```

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')
```

The following example adds a secured discussions server connection property called `securedProperty` with the value `secureValue`.

```
wls:/weblogic/serverConfig> setDiscussionForumConnectionProperty
(appName='webcenter', name='MyDiscussionServer', key='securedProperty',
value='secureValue', secure=1)
```

10.7.4 deleteDiscussionForumConnectionProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.4.1 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.

10.7.4.2 Syntax

```
deleteDiscussionForumConnectionProperty(appName, name, key, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.4.3 Example

The following example deletes a discussions server connection property named `myProperty1`.

```
wls:/weblogic/serverConfig> deleteDiscussionForumConnectionProperty
(appName='webcenter', name='MyDiscussionServer', key='myProperty1')
```

10.7.5 listDiscussionForumConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.5.1 Description

Lists all of the discussions server connections that are configured for a named WebCenter Portal application.

10.7.5.2 Syntax

```
listDiscussionForumConnections(appName, [verbose, name, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Valid options are `1` (true) and `0` (false). When set to `1`, `listDiscussionForumConnections` lists all of the discussions server connections that are configured for a WebCenter Portal application, along with their details. When set to `0`, only connection names are listed. This argument defaults to `0`.
name	Optional. Name of an existing discussions server connection. Use this argument to view connection details for a specific discussions server connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.5.3 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=1)
```

The following example lists connection details for a discussions server connection named `myDiscussionsServer`.

```
wls:/weblogic/serverConfig> listDiscussionForumConnections(appName='webcenter',
name='myDiscussionsServer')
```

10.7.6 listDefaultDiscussionForumConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.6.1 Description

Names the discussions server connection that the Discussions service and the Announcements service are using, in a named WebCenter Portal application. While

you can register multiple discussions server connections for a WebCenter Portal application, the Discussions/Announcements service only uses one connection—known as the default (or active) connection.

10.7.6.2 Syntax

```
listDefaultDiscussionForumConnection(appName, [verbose, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Valid options are `1` (true) and `0` (false). When set to `1`, the name and details of the discussions server connections are listed. When set to `0`, only the connection name displays. This argument defaults to `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.6.3 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=1)
```

10.7.7 setDefaultDiscussionForumConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.7.1 Description

Specifies the *default* discussions server connection for the Discussions service and the Announcements service, in a named WebCenter Portal application.

While you can register multiple discussions server connections with a WebCenter Portal application, the Discussions/Announcements services only uses one connection—this is known as the default (or active) connection.

10.7.7.2 Syntax

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

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing discussions server connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.7.3 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')
```

10.7.8 setDiscussionForumServiceProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.8.1 Description

Specifies default values for the Discussions service.

Configurable properties for the Discussions service are listed in Table 10–10, " Discussion Service Configuration Properties".

Table 10–10 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.

Table 10–10 (Cont.) Discussion Service Configuration Properties

Configuration Property	Description
`ForumGatewayManager.AUTO_START`	Communication through mail distribution lists can be published as discussion forum posts. This parameter starts or stops the gateway for this communication. For the Spaces application, the default value is `1`, which means that as soon as you configure mail server settings through administration, the gateway starts. Set this to `0`, and restart the managed server, to stop the gateway and disable this feature. For WebCenter Portal applications, the default value is `0`. Set this to `1`, and restart the managed server, to start the gateway and enable this feature.

10.7.8.2 Syntax

```
setDiscussionForumServiceProperty(appName, property, value, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.8.3 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')
```

10.7.9 removeDiscussionForumServiceProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.9.1 Description

Removes the current value that is set for a Discussions service property. Use this command to remove any of the properties listed in Table 10–10, " Discussion Service Configuration Properties".

Take care when using this command as removing values for these properties might cause unexpected behavior.

Note: Use this command syntax to disable the connection currently used for discussion and announcement services:

```
removeDiscussionForumServiceProperty('appName='webce
nter', property='selected.connection')
```

This command forces the `default` connection argument to `0`. See also, setDiscussionForumConnection.

10.7.9.2 Syntax

```
removeDiscussionForumServiceProperty(appName, property, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
property	Name of the configuration property.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.9.3 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')
```

10.7.10 listDiscussionForumServiceProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.10.1 Description

Lists all configurable properties for the Discussions service.

10.7.10.2 Syntax

```
listDiscussionForumServiceProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.10.3 Example

The following example lists configuration properties for the Discussions service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig>
listDiscussionForumServiceProperties(appName='webcenter')
```

10.7.11 setAnnouncementServiceProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.11.1 Description

Specifies default values for the Announcements service.

Configurable properties for the Announcements service are listed in Table 10–11, " Announcements Service Configuration Properties".

Table 10–11 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.

10.7.11.2 Syntax

```
setAnnouncementServiceProperty(appName, property, value, [server,
applicationVersion})
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
property	Name of the configuration property.
value	Property value.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.11.3 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')
```

10.7.12 removeAnnouncementServiceProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.12.1 Description

Removes the current value that is set for a Announcements service property. Use this command to remove any of the properties listed in Table 10–11, " Announcements Service Configuration Properties".

Take care when using this command as removing values for these properties might cause unexpected behavior.

10.7.12.2 Syntax

```
removeAnnouncementServiceProperty(appName, property, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
property	Name of the configuration property.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.12.3 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')
```

10.7.13 listAnnouncementServiceProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.13.1 Description

Lists all configurable properties for the Announcements service.

10.7.13.2 Syntax

```
listAnnouncementServiceProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.13.3 Example

The following example lists configuration properties for the Announcements service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig> listAnnouncementServiceProperties(appName='webcenter')
```

10.7.14 addDiscussionsServerAdmin

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.14.1 Description

Grants system administrator permissions on the discussions server to a user or a group. This command is useful when you connect the discussions server to a new identity store that does not contain any of the current administrators.

10.7.14.2 Syntax

```
addDiscussionsServerAdmin(appName, name, [type, server, applicationVersion])
```

Argument	Definition
appName	Name of the discussions server application in which to perform this operation. For example, `owc_discussions`.
name	Name of the user or group to add as an administrator on the discussions server.
type	Optional. Identifies the type of identity. Valid values are `USER` and `GROUP`. The default value is `USER`.
server	Optional. Name of the managed server on which the application is deployed. For example, `WC_Collaboration`. 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 application is deployed.

10.7.14.3 Example

The following example grants system administrator permissions on the discussions server to the user `weblogic`:

```
addDiscussionsServerAdmin(appName='owc_discussions', name='weblogic', type='USER')
```

The following example grants system administrator permissions on the discussions server to all users in the `Administrators` user group:

```
addDiscussionsServerAdmin(appName='owc_discussions', name='Administrators',
type='GROUP')
```

10.7.15 syncDiscussionServerPermissions

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.15.1 Description

(Spaces application only) Synchronizes discussion server permissions for subspaces that inherit security from their parent.

When you update Discussions or Announcement permissions for space hierarchies in the Spaces application, the subspaces do not automatically inherit the corresponding permission change on WebCenter Portal's discussions server. Therefore, whenever changes are made, you must run this command to synchronize Discussions and Announcement permissions within a space hierarchy, such that subspaces inherit the same discussions server permissions as their parent.

> **Note:** To execute discussion server WLST commands, such as `syncDiscussionServerPermissions`, the user used to connect to the Admin Server must also have administrative privileges on the discussion server.

10.7.15.2 Syntax

```
syncDiscussionServerPermissions(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which to perform this operation—always `webcenter`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.7.15.3 Example

The following example synchronizes Discussions and Announcement permissions in the Spaces application, that is, subspaces inherit the same discussions server permissions as their parent:

```
wls:/weblogic/serverConfig> syncDiscussionServerPermissions(appName='webcenter')
```

10.7.16 setDiscussionsServerProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.16.1 Description

Sets a discussions server property.

Use this command to set a system property on the discussions server.

> **Note:** To execute discussion server WLST commands, such as `setDiscussionsServerProperty`, the user used to connect to the Admin Server must also have administrative privileges on the discussion server.

10.7.16.2 Syntax

```
setDiscussionsServerProperty(appName, key, value, [server, applicationVersion])
```

Argument	Definition
appName	Name of the discussions server application in which to perform this operation. For example, `owc_discussions`.
key	Name of the discussions server property.
value	Value for the discussions server property.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Collaboration`. 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 application is deployed.

10.7.16.3 Example

The following example sets properties that configures the discussions server for SSO, where `example.com:8890/owc_discussions` is the base URL of the webtier on which the discussions server is deployed:

```
wls:/weblogic/serverConfig>
setDiscussionsServerProperty(appName='owc_discussions', key='owc_
discussions.sso.mode', value='true')
setDiscussionsServerProperty(appName='owc_discussions', key='jiveURL',
value='example.com:8890/owc_discussions')
```

10.7.17 getDiscussionsServerProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.17.1 Description

Returns the current value of a discussions server property.

Note: To execute discussion server WLST commands, such as `getDiscussionsServerProperty`, the user used to connect to the Admin Server must also have administrative privileges on the discussion server.

10.7.17.2 Syntax

```
getDiscussionsServerProperty(appName, key, [server, applicationVersion])
```

Argument	Definition
appName	Name of the discussions server application in which to perform this operation. For example, `owc_discussions`.
key	Name of the discussions server property. For example, `owc_discussions.sso.mode`, `AuthFactory.className`, `UserManager.className`, `GroupManager.className`, `owc_discussions.setup.complete_11.1.1.2.0`, and so on.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Collaboration`. 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 application is deployed.

10.7.17.3 Example

The following example returns current values for some key discussions server properties:

```
wls:/weblogic/serverConfig> getDiscussionsServerProperty
(appName='owc_discussions', key='AuthFactory.className')
getDiscussionsServerProperty
(appName='owc_discussions', key='UserManager.className')
getDiscussionsServerProperty
(appName='owc_discussions', key='GroupManager.className')
getDiscussionsServerProperty
(appName='owc_discussions', key=', )
```

10.7.18 removeDiscussionsServerProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.7.18.1 Description

Removes the current values that is set for a discussions server property.

Note: To execute discussion server WLST commands, such as `removeDiscussionsServerProperty`, the user used to connect to the Admin Server must also have administrative privileges on the discussion server.

10.7.18.2 Syntax

```
removeDiscussionsServerProperty(appName, key, [server, applicationVersion])
```

Argument	Definition
appName	Name of the discussions server application in which to perform this operation. For example, `owc_discussions`.
key	Name of the discussions server property. For example, `owc_discussions.sso.mode`, `AuthFactory.className`, `UserManager.className`, `GroupManager.className`, `owc_discussions.setup.complete_11.1.1.2.0`, and so on.
server	Optional. Name of the managed server where the application is deployed. For example, `WC_Collaboration`. 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 application is deployed.

10.7.18.3 Example

The following example removes the current value for the 'SSO mode' property on the discussions server:

```
wls:/weblogic/serverConfig> removeDiscussionsServerProperty
(appName='owc_discussions', key='owc_discussions.sso.mode')
```

10.8 External Applications

Use the commands listed in Table 10–12 to manage external application connections for WebCenter Portal applications.

Configuration changes made using these WebCenter Portal WLST commands are immediately available in the WebCenter Portal application.

Table 10–12 External Application WLST Commands

Use this command...	To...	Use with WLST...
createExtAppConnection	Create an external application connection, for a named WebCenter Portal application.	Online
setExtAppConnection	Edit an existing external application connection.	Online
listExtAppConnections	List individual or all external applications that are configured for a specific WebCenter Portal 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

10.8.1 createExtAppConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.8.1.1 Description

Creates an external application connection, for a named WebCenter Portal application.

10.8.1.2 Syntax

```
createExtAppConnection(appName, name, [displayName, url, authMethod,
userFieldName, pwdFieldName, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal application.
displayName	Optional. External application display name. A user friendly name for the application that WebCenter Portal users will recognize. The display name must be unique across all external applications within the WebCenter Portal 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` and a login `url` is specified. Not required 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` and a login `url` is specified. Not required if `BASIC` authentication method is selected.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.8.1.3 Example

The following example creates a connection for an external application named `My Yahoo!`, in a WebCenter Portal 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')
```

10.8.2 setExtAppConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.8.2.1 Description

Edits an existing external application connection.

10.8.2.2 Syntax

```
setExtAppConnection(appName, name, [displayName], [url], [authMethod],
[userFieldName], [pwdFieldName], [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal users will recognize. The display name must be unique across all external applications within the WebCenter Portal 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.
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` and a login URL is specified 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, `WC_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.

10.8.2.3 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!')
```

10.8.3 listExtAppConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.8.3.1 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.

10.8.3.2 Syntax

```
listExtAppConnections(appName, [verbose, name, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application for which to perform this operation.
verbose	Optional. Displays external application details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listExtAppConnections` lists all of the external applications that are configured for a WebCenter application, along with their details. When set to `0`, `listExtAppConnections` lists only the names of the external applications. This argument defaults to `0`. If you set this argument to `0`, 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.8.3.3 Examples

The following example lists the names of all the external applications currently used by a WebCenter Portal 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=1)
----
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: {Account1:1, Accout2:DefVal:0}
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=1,
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
```

10.8.4 addExtAppField

Module: Oracle WebCenter Portal

Use with WLST: Online

10.8.4.1 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 `addExtAppCredential` and

setExtAppCredential to update the shared/public credentials. See Section 10.8.7, "addExtAppCredential" and Section 10.8.8, "setExtAppCredential".

10.8.4.2 Syntax

```
addExtAppField(appName, name, fieldName, [fieldValue], [displayToUser], [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which 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 1 (true) and 0 (false). This argument defaults to 0. Note that if you set this argument to 0, you must specify the fieldValue.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.8.4.3 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=1)
```

10.8.5 setExtAppField

Module: Oracle WebCenter Portal

Use with WLST: Online

10.8.5.1 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 1,

you may also need to update existing shared user or public user credentials. See also Section 10.8.8, "setExtAppCredential".

10.8.5.2 Syntax

```
setExtAppField(appName, name, fieldName, [fieldValue], [displayToUser], [server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 `1` (true) and `0` (false). If set to `0`, `fieldValue` must be specified.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.8.5.3 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=1)
```

10.8.6 removeExtAppField

Module: Oracle WebCenter Portal

Use with WLST: Online

10.8.6.1 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 10.8.8, "setExtAppCredential"

10.8.6.2 Syntax

```
removeExtAppField(appName, name, fieldName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.8.6.3 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')
```

10.8.7 addExtAppCredential

Module: Oracle WebCenter Portal

Use with WLST: Online

10.8.7.1 Description

Configures shared user or public user credentials for a specific external application.

When shared credentials are specified, every user accessing the WebCenter Portal application is authenticated using the user name and password defined here. WebCenter Portal users are not presented with a login form.

Public users accessing this application through WebCenter Portal are logged in using the public credentials defined here.

If credentials already exists, a warning indicates that the `setExtAppCredential` command should be used instead.

10.8.7.2 Syntax

```
addExtAppCredential(appName, name, type, username, password, [field, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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.
field	Optional. Additional login field value. Use the format `FieldName:FieldValue`, where `FieldName` names an additional login field configured with `displayToUser=1`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.8.7.3 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')
```

10.8.8 setExtAppCredential

Module: Oracle WebCenter Portal

Use with WLST: Online

10.8.8.1 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 10.8.7, "addExtAppCredential".

The arguments `username` and `password` are optional because `setExtAppCredential` only manipulates existing credentials. At least one of the parameters, `username`, `password` or `field`, must be specified.

You can use `setExtAppCredential` command to update passwords in systems that require changing passwords every few days.

10.8.8.2 Syntax

```
setExtAppCredential(appName, name, type, [username], [password], [field],
[server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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=1`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.8.8.3 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')
```

10.8.9 removeExtAppCredential

Module: Oracle WebCenter Portal

Use with WLST: Online

10.8.9.1 Description

Removes shared user or public user credentials currently configured for an external application.

If credentials do not exist, an error displays.

10.8.9.2 Syntax

```
removeExtAppCredential(appName, name, type, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.8.9.3 Example

The following example removes shared credentials specified for an external application named ABC.

```
wls:/weblogic/serverConfig> removeExtAppCredential(appName='webcenter',
name='ABC', type='SHARED')
```

10.9 Instant Messaging and Presence

Use the commands listed in Table 10–13, to manage instant messaging and presence server connections.

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

Table 10–13 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 Portal 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 Portal 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
createIMPExtAppConnection	Create an external application suitable for instant messaging and presence server connection.	Online

10.9.1 createIMPConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.1.1 Description

Creates an instant messaging and presence server connection for a named WebCenter Portal application.

Use the listIMPAdapters command to find out which types of instant messaging and presence servers are supported. Out-of-the-box, WebCenter Portal applications support Microsoft Live Communications Server 2005 (LCS), and Microsoft Communications Server 2007 (OCS), and Microsoft Lync 2010.

While you can register multiple presence server connections for a WebCenter Portal application, only one connection is used for instant messaging and presence services—the default (or active) connection.

10.9.1.2 Syntax

```
createIMPConnection(appName, name, adapter, url, [appId, poolName,
userDomain, timeout, default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal application.
adapter	Adapter name. Specify the adapter that matches your instant messaging and presence server. Valid values are `LCS` and `OCS` Choose `LCS` for Microsoft Live Communications Server 2005. Choose `OCS2007` for Microsoft Office Communications Server 2007 and Microsoft Lync.
url	URL of the sever hosting instant messaging and presence services. For example: `http://myocshost.com:8888`
domain	Deprecated. Use the setIMPServiceProperty command to resolve IM addresses.
appId	Optional. External application associated with the presence server connection. If specified, external application credential information is used to authenticate users against the LCS, OCS, or Lync server. This argument is mandatory for LCS, OCS and Lync server connections. The external application you configure for instant messaging and presence services must use `authMethod=POST`, and specify an additional field with `fieldName='Account'` and `displaytoUser=1`. If an external application does not exist yet, use the WLST command createIMPExtAppConnection to create an external application that automatically has all the required additional fields. See also addExtAppField and setExtAppField.

Argument	Definition
poolName	Optional. Pool name that is required to create an LCS, OCS, or Lync connection. Refer to *Microsoft Live Communications Server*, *Microsoft Office Communications Server*, or Microsoft Lync Server documentation for details on pool names. This argument is mandatory for LCS, OCS, and Lync server connections.
userDomain	Optional. (OCS and Lync connections only.) Active Directory domain on the OCS/Lync server. This argument is mandatory for OCS/Lync server connections.
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.
default	Optional. Indicates whether this connection is the default connection for the Instant Messaging and Presence service. Valid values are `1` (true) and `0` (false). The default for this argument is `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.1.3 Examples

The following example creates an external application suitable for an instant messaging and presence server connection and then creates a connection named `myLCSPresenceServer` to a Microsoft Live Communications Server:

```
wls:/weblogic/serverConfig> createIMPExtApp(appName='webcenter', name='LCSExtApp',
displayName='IMP Ext App')
wls:/weblogic/serverConfig> createIMPConnection(appName='webcenter',
name='myLCSPresenceServer', adapter='LCS', url='http://mylcshost.com/owc/lcs',
appId='LCSExtApp', poolName='pool1.myhost.com', timeout=60,
default=1)
```

The following example creates an instant messaging and presence server connection to a Microsoft Office Communications Server named `myOCSPresenceServer`.

```
wls:/weblogic/serverConfig> createIMPConnection(appName='webcenter',
name='myOCSPresenceServer', adapter='OCS2007', url='http://myocshost.com/owc/ocs',
appId='OCSExtApp', userDomain='OCS', poolName='pool01.myocshost.com', timeout=60,
default=1)
```

The following example creates an instant messaging and presence server connection to a Microsoft Lync Server named `myLyncServer`.

```
wls:/weblogic/serverConfig>createIMPConnection(appName='webcenter',
name='myLyncServer', adapter='OCS2007',
url='http://mylynchost.com:8888' appId='LyncExtApp', userDomain='LYNC',
poolName='pool05.mylynchost.com', timeout=60, default=1))
```

10.9.2 setIMPConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.2.1 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.

10.9.2.2 Syntax

```
setIMPConnection(appName, name, [adapter, url, appId, poolName,
userDomain, timeout, default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
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 `OCS2007`. Choose `LCS` for Microsoft Live Communications Server. Choose `OCS2007` for Microsoft Office Communications Server and Microsoft Lync Server.
url	Optional. URL of the server hosting instant messaging and presence services.
domain	Deprecated. Use the setIMPServiceProperty command to resolve IM addresses.
appId	Optional. External application associated with the presence server connection. If specified, external application credential information is used to authenticate users against the LCS, OCS, or Lync server. This argument is mandatory for LCS, OCS and Lync server connections. The external application you configure for instant messaging and presence services must use `authMethod=POST`, and specify an additional field with `fieldName='Account'` and `displaytoUser=1`. If an external application does not exist yet, use the WLST command createIMPExtAppConnection to create an external application that automatically has all the required additional fields. See also addExtAppField and setExtAppField.
poolName	Optional. (LCS, OCS, and Lync) Pool name that is required to create an LCS, OCS, or Lync connection. Refer to *Microsoft Live Communications Server*, *Microsoft Office Communications Server* or Microsoft Lync Server documentation for details on pool names. This argument is mandatory for LCS, OCS, and Lync server connections.
userDomain	Optional. (OCS and Lync only.) Active Directory domain on the OCS server. This argument is mandatory for OCS/Lync server connections.

Argument	Definition
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.
default	Optional. Indicates whether this connection is the default connection for the Instant Messaging and Presence service. Valid values are `1` (true) and `0` (false). The default for this argument is `0`. To specify that the Instant Messaging and Presence service uses this connection, change the value from `0` to `1`. 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 Portal 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.2.3 Examples

The following example sets attributes on an existing instant messaging and presence server connection.

```
wls:/weblogic/serverConfig>setIMPConnection(appName='webcenter',
name='myOCSPresenceServer', adapter='OCS2007', url='http://myocshost.com/owc/ocs',
timeout=120, default=0)
```

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',
appId='LCSExtApp', poolName='pool3.myhost.com')
```

10.9.3 setIMPConnectionProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.3.1 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.)

Do not use the setIMPConnectionProperty to set connection properties available through createIMPConnection or setIMPConnection. Attempting to do so, has no effect.

10.9.3.2 Syntax

```
setIMPConnectionProperty(appName, name, key, value, [secure, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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.
secure	Optional. Indicates whether the property value must be stored securely using encryption. Valid options are `1` (true) and `0` (false). When `1`, the value is encrypted. The default option is `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.3.3 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')
```

10.9.4 deleteIMPConnectionProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.4.1 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.

10.9.4.2 Syntax

```
deleteIMPConnectionProperty(appName, name, key, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.4.3 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')
```

10.9.5 listIMPAdapters

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.5.1 Description

Lists which types of instant messaging and presence servers WebCenter Portal supports. Out-of-the-box, WebCenter Portal applications support Microsoft Live Communications Server 2005 (LCS), Microsoft Office Communications Server 2007 (OCS) and Microsoft Lync 2010.

10.9.5.2 Syntax

```
listIMPAdapters()
```

10.9.5.3 Example

The following example lists which presence servers are supported.

```
wls:/weblogic/serverConfig> listIMPAdapters()
```

10.9.6 listIMPConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.6.1 Description

Lists all of the instant messaging and presence server connections that are configured for a named WebCenter Portal application.

10.9.6.2 Syntax

```
listIMPConnections(appName,[verbose], [name], [server, applicationVersion)
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Displays presence server connection details in verbose mode. Valid values are `1` (true) and `0` (false). When set to `1`, `listIMPConnections` lists all of the presence server connections that are configured for a WebCenter Portal application, along with their details. When set to `0`, only connection names are listed. This argument defaults to `0`.
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 `1`, the `verbose` argument is ignored.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.6.3 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=1)
```

The following example lists connection details for an instant messaging and presence server connections named `impConnection1`.

```
wls:/weblogic/serverConfig> listIMPConnections(appName='webcenter',
name='impConnection1')
```

10.9.7 listDefaultIMPConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.7.1 Description

Lists the connection that the Instant Messaging and Presence service is using, in a named WebCenter Portal application. While you can register multiple presence server connections for a WebCenter Portal 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.

10.9.7.2 Syntax

```
listDefaultIMPConnection(appName, verbose, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Displays the default presence server connection in verbose mode, if available. Valid options are `1` (true) and `0` (false). When set to `1`, the name and details of the presence server connection are listed. When set to `0`, only the connection name displays. This argument defaults to `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.7.3 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=1)
```

10.9.8 setDefaultIMPConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.8.1 Description

Specifies the *default* connection for the Instant Messaging and Presence service, in a named WebCenter Portal application. While you can register multiple presence server connections with a WebCenter Portal 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.

10.9.8.2 Syntax

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

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing instant messaging and presence connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.8.3 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')
```

10.9.9 setIMPServiceProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.9.1 Description

Specifies default values for the Instant Messaging and Presence service.

Configurable properties for the Instant Messaging and Presence service are listed in Table 10–14, " Instant Messaging and Presence Service Configuration Properties".

Table 10–14 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 `0`, and display name information is unavailable, only the user name displays in the application. When set to `1`, and display name information is unavailable, display names are read from user profile data. Setting this option to `1` will impact performance. The default setting is 0. Display names are not mandatory in presence data. If the WebCenter Portal application does not always provide display names by default and you consider this information important, set `resolve.display.name.from.user.profile` to `1` so that display names always display.
im.address.resolver.class	Resolver implementation used to map user names to IM addresses and IM addresses to user names. The default setting is `oracle.webcenter.collab.rtc.IMPAddressResolverImpl`. This implementation looks for IM addresses in the following places and in the order specified: ▪ User Preferences ▪ User Credentials ▪ User Profiles
im.address.profile.attribute	User profile attribute used to determine a user's IM address. The default setting is `BUSINESS_EMAIL`.

10.9.9.2 Syntax

```
setIMPServiceProperty(appName, property, value, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.9.3 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')
```

10.9.10 removeIMPServiceProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.10.1 Description

Removes the current value that is set for an Instant Messaging and Presence service property. Use this command to remove any of the properties listed in Table 10–14, " Instant Messaging and Presence Service Configuration Properties".

Take care when using this command as removing values for these properties 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 `0`. See also, setIMPConnection.

10.9.10.2 Syntax

```
removeIMPServiceProperty(appName, property, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

Argument	Definition
property	Name of the configuration property.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.10.3 Example

The following example clears the 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')
```

10.9.11 listIMPServiceProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.11.1 Description

Lists all configurable properties for the Instant Messaging and Presence service.

10.9.11.2 Syntax

```
listIMPServiceProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.11.3 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')
```

10.9.12 createIMPExtAppConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.9.12.1 Description

Creates an external application suitable for instant messaging and presence server connections. The external application is configured with the required additional properties: `authMethod=POST`, and additional fields `fieldName='Account'` and `displaytoUser=1`.

10.9.12.2 Syntax

```
createIMPExtAppConnection(appName, name, [displayName, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal application.
displayName	Optional. External application display name. A user friendly name for the application that WebCenter Portal users will recognize. The display name must be unique across all external applications within the WebCenter Portal application.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.9.12.3 Example

The following example creates an external application named IMPxApp suitable for instant messaging and presence server connections.

```
wls:/weblogic/serverConfig> createIMPExtAppConnection(appName='webcenter',
name='IMPxApp', displayName='IMP Ext App')
```

10.10 Mail

Use the commands listed in Table 10–15 to manage mail server connections for a WebCenter Portal application.

You can register multiple mail server connections:

- **Spaces application** supports multiple mail connections. The mail connection configured with `default=1` is the default connection for mail services in the Spaces application. All additional connections are offered as alternatives; Spaces users can choose which one they want to use through user preferences.
- **Framework applications** only use one mail connection—the connection configured with `default=1`. Any additional connections are ignored.

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

Table 10–15 Mail WLST Commands

Use this command...	To...	Use with WLST...
createMailConnection	Create a mail server connection for a WebCenter Portal 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
removeMailServiceProperty	Remove defaults for the Mail service.	Online
listMailServiceProperties	List Mail service properties.	Online
createMailExtApp	Create an external application suitable for mail connections.	Online

10.10.1 createMailConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.1.1 Description

Creates a mail server connection for a WebCenter Portal application.

WebCenter Portal 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:

- **Spaces application** supports multiple mail connections. The mail connection configured with `default=1` is the default connection for mail services in Spaces. All additional connections are offered as alternatives; Spaces users can choose which one they want to use through user preferences.
- **Framework applications** only use one mail connection—the connection configured with `default=1`. Any additional connections are ignored.

10.10.1.2 Syntax

```
createMailConnection(appName, name, [imapHost, imapPort, smtpHost, smtpPort,
imapSecured, smtpSecured, appId, timeout, default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal application.
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 mail server connection to the IMAP server is SSL-enabled. Valid values are `1` (true) and `0` (false). The default for this argument is `0`.
smtpSecured	Optional. Specifies whether the SMTP server is secured. Valid values are `1` (true) and `0` (false). The default for this argument is `0`.

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.
	The external application you configure for the Mail service must use `authMethod=POST`, and specify several additional login fields:
	`fieldName='Email Address'` and `displaytoUser=1`
	`fieldName='Your Name'` and `displaytoUser=1`
	`fieldName='Reply-To Address'` and `displaytoUser=1`
	If an external application does not exist yet, use the WLST command createMailExtApp to create an external application that automatically has all the required additional fields.
	See also createExtAppConnection.

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 `1` (true) and `0` (false). This argument defaults to `0`. ■ **Spaces** supports multiple mail connections. The mail connection configured with `default=1` is the default connection for mail services in the Spaces application. Additional connections, configured with `default=0`, are offered as alternatives; Spaces users can choose which one they want to use through user preferences. ■ **Framework applications** only use one mail connection—the connection configured with `default=1`. Any additional connections are ignored.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.1.3 Examples

The following example creates an external application suitable for a mail server connection, and then creates a mail server connection named `myMailConnection`:

```
wls:/weblogic/serverConfig> createMailExtApp(appName='webcenter', name='extApp_
Mail', displayName='Mail Ext App')
wls:/weblogic/serverConfig> createMailConnection(appName='webcenter' ,
name='myMailConnection' , imapHost='myimaphost.com', imapPort=143 ,
smtpHost='mysmtphost.com' , smtpPort=25 , imapSecured=0, smtpSecured=0,
appId='extApp_Mail', timeout=60, default=1)
```

10.10.2 setMailConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.2.1 Description

Edits an existing mail connection. Use this command to update connection attributes.

The connection is created using the createMailConnection command.

(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 spacedistribution lists for the Spaces application. Distribution lists are named after their space (excluding non-java identifiers) and assigned a domain (derived from the `domain` attribute, for example, `@mycompany.com`). If LDAP details are not provided, spacedistribution lists are not created or maintained. The mail server must be a *Microsoft Exchange Server*.

10.10.2.2 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 Portal application in which 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.

Argument	Definition
imapSecured	Optional. Specifies whether the connection to the IMAP server is secured (SSL-enabled). Valid values are `1` (true) and `0` (false). The default for this argument is `0`.
smtpSecured	Optional. Specifies whether the connection to the SMTP server is secured (SSL-enabled). Valid values are `1` (true) and `0` (false). The default for this argument is `0`.

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.
	The external application you configure for the Mail service must use `authMethod=POST`, and specify several additional login fields:
	`fieldName='Email Address'` and `displaytoUser=1`
	`fieldName='Your Name'` and `displaytoUser=1`
	`fieldName='Reply-To Address'` and `displaytoUser=1`
	If an external application does not exist yet, use the WLST command createMailExtApp to create an external application that automatically has all the required additional fields.
	See also createExtAppConnection.

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.
ldapSecured	Optional. Specifies whether the connection to the LDAP server is secured (SSL enabled). Valid values are `1` (true) and `0` (false). The default for this argument is `0`. Set this to `1` for all LDAP communications over SSL.
domain	Optional. Domain name appended to spacedistribution lists. For example, if the domain attribute is set to `mycompany.com`, the `Finance Project` 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 spacedistribution 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.

Argument	Definition
default	Optional. Indicates whether this connection is the default (or active) connection for the Mail service. Valid values are `1` (true) and `0` (false). This argument defaults to `0`. `1` specifies that this connection is the default connection for the Mail service. ▪ **Spaces** supports multiple mail connections. The mail connection configured with `default=1` is the default connection for mail services in the Spaces application. Additional connections, configured with `default=0`, are offered as alternatives; Spaces users can choose which one they want to use through user preferences. ▪ **Framework applications** only use one mail connection—the connection configured with `default=1`. 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 `0` to `1`. 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.2.3 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=0, smtpSecured=0,
appId='extApp_Mail', timeout=60, default=1)
```

The following example sets individual attributes of a mail server connection.

```
wls:/weblogic/serverConfig>setMailConnection(appName='webcenter',
name='myMailConnection', imapPort=993, imapSecured=1, smtpPort=465 ,
smtpSecured=1)
```

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=0, smtpSecured=0, smtpPort=25, appId='extApp_Mail',
default=1, ldapHost='myldaphost.com', ldapPort=389,
ldapBaseDN='CN=Users,DC=exchange,DC=uk,DC=com', ldapAdminUser='administrator',
ldapAdminPassword='adminpswd', ldapSecured=0, timeout=60)
```

10.10.3 setMailConnectionProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.3.1 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 10–16, " Additional Mail Connection Properties".

Table 10–16 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.

10.10.3.2 Syntax

```
setMailConnectionProperty(appName, name, key, value, [secure], [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter application in which 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 `1` (true) and `0` (false). When `1`, the value is encrypted. The default option is `0`.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WC_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.

10.10.3.3 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')
```

10.10.4 deleteMailConnectionProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.4.1 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.

10.10.4.2 Syntax

```
deleteMailConnectionProperty(appName, name, key, [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.4.3 Example

The following example deletes a mail server connection property named `mailProperty1`.

```
wls:/weblogic/serverConfig> deleteMailConnectionProperty(appName='webcenter',
name='myMailServer', key='mailProperty1')
```

10.10.5 listMailConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.5.1 Description

Lists all of the mail server connections that are configured for a named WebCenter Portal application.

10.10.5.2 Syntax

```
listMailConnection(appName, [verbose, name, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Displays mail server connection details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listMailConnections` lists all of the mail server connections that are configured for a WebCenter Portal application, along with their details. When set to `0`, only connection names are listed. This argument defaults to `0`.
name	Optional. Name of an existing mail server connection. Use this argument to view connection details for a specific mail server connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.5.3 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=1)
```

The following example lists connection details for a mail server connection named `mailConnection1`.

```
wls:/weblogic/serverConfig> listMailConnections(appName='webcenter',
name='mailConnection1')
```

10.10.6 listDefaultMailConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.6.1 Description

Lists the default mail server connection that the Mail service is using, in a named WebCenter Portal application.

You can register multiple mail server connections but there can only be one default connection:

- **Spaces application** supports multiple mail connections. The mail connection configured with `default=1` is the default connection for mail services in the

Spaces application. All additional connections are offered as alternatives; Spaces users can choose which one they want to use through user preferences.

- **Framework applications** only use one mail connection—the connection configured with `default=1`. Any additional connections are ignored.

10.10.6.2 Syntax

```
listDefaultMailConnection(appName,[verbose], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Displays the default mail server connection in verbose mode, if available. Valid options are `1` (true) and `0` (false). When set to `1`, the name and details of the mail server connection are listed. When set to `0`, only the connection name displays. This argument defaults to `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.6.3 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=1)
```

10.10.7 setDefaultMailConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.7.1 Description

Specifies the *default* mail server connection for the Mail service, in a named WebCenter Portal application.

You can register multiple mail server connections but there can only be one default connection:

- **Spaces application** supports multiple mail connections. The mail connection configured with `default=1` is the default connection for mail services in the Spaces application. All additional connections are offered as alternatives; Spaces users can choose which one they want to use through user preferences.
- **Framework applications** only use one mail connection—the connection configured with `default=1`. Any additional connections are ignored.

10.10.7.2 Syntax

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

Argument	Description
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing mail connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.7.3 Example

The following example configures the Mail service to use a connection named `myMailServer`.

```
wls:/weblogic/serverConfig>setDefaultMailConnection(appName='webcenter',
name='myMailServer')
```

10.10.8 setMailServiceProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.8.1 Description

Specifies default values for the Mail service.

Configurable properties for the Mail service are listed in Table 10–17, " Mail Service Configuration Properties".

Table 10–17 Mail Service Configuration Properties

Configuration Property	Description
address.delimiter	Defines the delimiter that is used to separate multiple mail addresses. A comma is used by default. Some mail servers require mail addresses in the form `lastname, firstname` and, in such cases, a semi-colon is required.
mail.emailgateway.polling.frequency	The frequency, in seconds, that spacedistribution lists are checked for new incoming emails. The default is 1800 seconds (30 minutes). Email communication through spacedistribution lists can be published as discussion forum posts on a discussions server. For details, see "Publishing Space Mail in a Discussion Forum" in *Oracle Fusion Middleware User's Guide for Oracle WebCenter*.
mail.messages.fetch.size	Maximum number of messages displayed in mail inboxes.

Table 10–17 (Cont.) Mail Service Configuration Properties

Configuration Property	Description
resolve.email.address.to.name	Determines whether user email addresses are resolved to WebCenter Portal user names when LDAP is configured. Valid values are `1` (true) and `0` (false). The default value is `0`. When set to `1`, WebCenter Portal user names display instead of email addresses in Mail task flows. Set this property to `1` if the Instant Messaging and Presence service requires user names to obtain presence status because presence information cannot be obtained when the Mail service provides email addresses. Setting this value to `1` does impact application performance so you must take this into consideration when setting this property.

10.10.8.2 Syntax

```
setMailServiceProperty(appName, property, value, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.8.3 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')
```

10.10.9 removeMailServiceProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.9.1 Description

Removes the current value that is set for a Mail service property. Use this command to remove any of the properties listed in Table 10–17, " Mail Service Configuration Properties".

Take care when using this command as removing values for these properties 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 `0`. See also, setMailConnection.

10.10.9.2 Syntax

```
removeMailServiceProperty(appName, property, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
property	Name of the configuration property.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.9.3 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')
```

10.10.10 listMailServiceProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.10.1 Description

Lists all configurable properties for the Mail service.

10.10.10.2 Syntax

```
listMailServiceProperties(appName, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.10.3 Example

The following example lists configuration properties for the Mail service, in an application named `webcenter`.

```
wls:/weblogic/serverConfig>listMailServiceProperties(appName='webcenter')
```

10.10.11 createMailExtApp

Module: Oracle WebCenter Portal

Use with WLST: Online

10.10.11.1 Description

Creates an external application suitable for mail server connections. The external application is configured with the required additional properties: `authMethod=POST`, and specify several additional login fields:

`fieldName='Email Address'` and `displaytoUser=1`

`fieldName='Your Name'` and `displaytoUser=1`

`fieldName='Reply-To Address'` and `displaytoUser=1`

10.10.11.2 Syntax

```
createMailExtAppConnection(appName, name, [displayName, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
`name`	Connection name. The name must be unique (across all connection types) within the WebCenter Portal application.
displayName	Optional. External application display name. A user friendly name for the application that WebCenter Portal users will recognize. The display name must be unique across all external applications within the WebCenter Portal application.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.10.11.3 Example

The following example creates an external application named `MailxApp` suitable for mail server connections.

```
wls:/weblogic/serverConfig> createMailExtAppConnection(appName='webcenter',
name='MailxApp', displayName='Mail Ext App')
```

10.11 Notifications

Use the commands listed in Table 10–18 to manage settings for the Notifications service in a WebCenter Portal application.

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

Table 10–18 Notifications WLST Commands

Use this command...	To...	Use with WLST...
setNotificationsConfig	Specify the connection used for routing notifications raised in a WebCenter Portal application.	Online
getNotificationsConfig	Return details about the connection that is used to send notifications raised in a WebCenter Portal application.	Online

10.11.1 setNotificationsConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.11.1.1 Description

Specifies the connection used for routing notifications raised in a WebCenter Portal application.

Use an existing mail server or BPEL server connection. If the WebCenter Portal application is connected to a BPEL server, the Oracle User Messaging Service (UMS) is available for routing notifications through multiple messaging channels, including mail, worklists, and SMS. If you configure the Notifications service to use a BPEL server connection, you may specify a sender 'From' address for each available messaging channel. That is, you can specify a sender mail address and an SMS address.

Alternatively, you can route notification messages through a mail server. If you configure the Notifications service to use a mail server connection, the external application associated with the mail server connection must contain shared credentials. Shared credentials are required for routing application-wide notifications.

10.11.1.2 Syntax

```
setNotificationsConfig(appName, type, name, [senderMailAddress, senderSMSAddress,
server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.

Argument	Definition
type	Type of connection used to send notifications. Valid values are `MAIL` and `BPEL`.
name	Name of an *existing* connection. Consider the following: ■ **Mail server connection**—The external application associated with the mail server connection must contain shared credentials. ■ **BPEL server connection**—Oracle User Messaging Service (UMS) must be available on the BPEL server.
senderMailAddress	Optional. Mail address from which all mail notifications are sent. Use the format: `<email_alias><<email_address>>` or `<email address>`. For example, `WebCenter Notification<notifications@webcenter.com>` or `notifications@webcenter.com`. This argument applies to notifications routed through BPEL servers. When a BPEL server is used and UMS is configured with multiple email drivers, this address is also used to identify the appropriate email driver. When a mail server is used, the "From Address" is the same user that is specified for the associated external application's shared credentials.
senderSMSAddress	Optional. SMS number from which all SMS notifications are sent. Typically, the SMS address format is a 4-6 digit number (without -, spaces, or any other characters). For example, `28734`. This argument applies to notifications routed through BPEL servers. When a BPEL server is used and UMS is configured with multiple SMS drivers, this address is also used to identify the appropriate SMS driver.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.11.1.3 Example

The following example specifies that the Notifications service use a BPEL server connection named 'WebCenter-Worklist' and also defines the mail address and SMS address from which all notifications are sent:

```
wls:/weblogic/serverConfig>setNotificationsConfig(appName='webcenter',
type='BPEL',
name='WebCenter-Worklist', senderMailAddress='WebCenter
Notification<notifications@webcenter.com',
senderSMSAddress='28734')
```

10.11.2 getNotificationsConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.11.2.1 Description

Returns details about the connection that is used to send notifications raised in a WebCenter Portal application.

10.11.2.2 Syntax

```
getNotificationsConfig(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.11.2.3 Example

The following example returns details about the connection used by the Notifications service in the Spaces application:

```
wls:/weblogic/serverConfig>getNotificationsConfig(appName='webcenter')

ConnectionType:    BPEL
ConnectionName:    WebCenter-Worklist
SenderMailAddress: notifications@webcenter.com
SenderSMSAddress:  28776
```

10.12 Personal Events

Use the commands listed in Table 10–19 to manage personal events server connections for a WebCenter Portal application.

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

Table 10–19 Personal Events WLST Commands

Use this command...	To...	Use with WLST...
createPersonalEventConnection	Create a personal events server connection for a named WebCenter Portal application.	Online
setPersonalEventConnection	Edit an existing personal events server connection.	Online
listPersonalEventConnections	List all of the personal events server connections that are configured for a named WebCenter Portal application	Online

10.12.1 createPersonalEventConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.12.1.1 Description

Creates a personal events server connection for a named WebCenter Portal application.

The Personal Events service supports connections to Microsoft Exchange Server 2003 and Microsoft Exchange Server 2007.

While you can register multiple personal events connections for a WebCenter Portal application, only one connection is used for personal events services - the default (or active) connection.

10.12.1.2 Syntax

```
createPersonalEventConnection(appName, name, webServiceUrl, adapterName, appId,
[default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
name	Connection name. The name must be unique (across all connection types within the WebCenter Portal application.
webServiceUrl	URL of the Web service exposing the event application. Use the format `<protocol>://<host>:<port>/<appWebServiceInterface>/<WSName>`
adapterName	Specify the adapter that matches the personal events server. Valid values are `MSEx2003` and `MSEx2007`. Choose `MSEx2003` for Microsoft Exchange Server 2003 and `MSEx2007` for Microsoft Exchange Server 2007. Each adapter has its own mechanism of authenticating and exchanging data
appId	External application associated with the Microsoft Exchange Server providing personal events services. If specified, external application credential information is used to authenticate users against the Microsoft Exchange Server.
default	Optional. Indicates whether this connection is the default connection for the Personal Events service. Valid values are `1` (true) and `0` (false). The default for this argument is `0`. To specify that the Personal Events service uses this connection, set the value to `1`. While you can register multiple connections for a WebCenter application, only one connection is used for personal event services—the default (or active) connection.
server	Optional. Name of the managed server where the WebCenter application is deployed. For example, `WC_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.

10.12.1.3 Example

The following example creates a connection named `MyPEConnection` for the Spaces application (`appName='webcenter'`). The connection points to a Microsoft Exchange Server 2007 and is designated as the default connection for the Personal Events service.

```
wls:/weblogic/serverConfig>createPersonalEventConnection(appName='webcenter',
name='MyPEConnection', webServiceUrl='http://myexchange.com/EWS/Services.wsdl',
adapterName='MSEx2007', appId='ExtPEApp', default=1)
```

The following example creates a connection named `MyPEConnection` for a Spaces application. The connection points to a Microsoft Exchange Server 2003.

```
wls:/weblogic/serverConfig>createPersonalEventConnection(appName='webcenter',
name='MyPEConnection',webServiceUrl='http://myexchange.com/ExchangeWS/PersonalEven
tsWebService.asmx', adapterName='MSEx2003',  appId='ExtPEApp')
```

10.12.2 setPersonalEventConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.12.2.1 Description

Edits a personal events server connection for a named WebCenter Portal application.

10.12.2.2 Syntax

```
setPersonalEventConnection(appName, name, [webServiceUrl, adapterName, appId,
default, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
name	Name of an existing personal events server connection.
webServiceUrl	Optional. URL of the Web service exposing the event application. Use the format `<protocol>://<host>:<port>/<appWebServiceInterface>/<WSName>`
adapterName	Optional. Specify the adapter that matches the personal events server. Valid values are `MSEx2003` and `MSEx2007`. Choose `MSEx2003` for Microsoft Exchange Server 2003 and `MSEx2007` for Microsoft Exchange Server 2007. Each adapter has its own mechanism of authenticating and exchanging data
appId	Optional. External application associated with the Microsoft Exchange Server providing personal events services. If specified, external application credential information is used to authenticate users against the Microsoft Exchange Server.

Argument	Definition
default	Optional. Indicates whether this connection is the default connection for the Personal Events service. Valid values are `1` (true) and `0` (false). The default for this argument is `0`. To specify that the Personal Events service uses this connection, set the value to `1`. While you can register multiple connections for a WebCenter Portal application, only one connection is used for personal event services—the default (or active) connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.12.2.3 Example

The following example updates the Web service URL for a connection named `MyPEConnection`.

```
wls:/weblogic/serverConfig>setPersonalEventConnection(appName='webcenter',
name='MyPEConnection', webServiceUrl='http://myexchange.com/EWS/Services.wsdl')
```

The following example makes a connection named `MyPEConnection` the default connection for personal events services in the Spaces application.

```
wls:/weblogic/serverConfig>setPersonalEventConnection(appName='webcenter',
name='MyPEConnection', default=1)
```

10.12.3 listPersonalEventConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.12.3.1 Description

Lists all of the personal events server connections that are configured for a named WebCenter Portal application.

10.12.3.2 Syntax

```
listPersonalEventConnections(appName, [verbose, name, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
verbose	Optional. Displays connection details for the Personal Events service in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listPersonalEventConnections` lists all of the personal events server connections that are configured for a WebCenter Portal application, along with their details. When set to `0`, only connection names are listed. This argument defaults to `0`. When set to `0`, do not specify the `name` argument.

Argument	Definition
name	Optional. Name of an existing personal events connection. Use this argument to view connection details for a specific personal events server.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.12.3.3 Example

The following example lists connection names and details for all of the personal events server connections currently configured for the Spaces application.

```
wls:/weblogic/serverConfig>listPersonalEventConnections(appName='webcenter',
verbose=1)
```

The following example displays connection details for a personal events server connection named `MyPEConnection`.

```
wls:/weblogic/serverConfig>listPersonalEventConnections(appName='webcenter',
verbose=1, name='MyPEConnection')
```

10.13 Personalization

Use the commands listed in Table 10–20 to manage personalization connections for a WebCenter Portal application.

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

Table 10–20 Personalization WLST Commands

Use this command...	To...	Use with WLST...
createWCPSCMISConnection	Create a CMIS connection for the WebCenter Portal's Personalization service.	Online
createWCPSActivityGraphConnection	Create an Activity Graph connection for the WebCenter Portal's Personalization service.	Online
createWCPSPeopleConnection	Create a People connection for the WebCenter Portal's Personalization service.	Online
createWCPSCustomConnection	Create a custom connection for the WebCenter Portal's Personalization service.	Online
setWCPSConnectionProperty	Modify properties of an existing connection for the WebCenter Portal's Personalization service.	Online
listWCPSCMISConnection	List CMIS connections configured for the WebCenter Portal's Personalization service.	Online
listWCPSActivityGraphConnection	List Activity Graph connections configured for the WebCenter Portal's Personalization service.	Online

Table 10–20 (Cont.) Personalization WLST Commands

Use this command...	To...	Use with WLST...
listWCPSPeopleConnection	List People connections configured for the WebCenter Portal's Personalization service.	Online
listWCPSCustomConnection	List custom connections configured for the WebCenter Portal's Personalization service.	Online
deleteWCPSCMISConnection	Create a CMIS connection for the WebCenter Portal's Personalization service.	Online
deleteWCPSActivityGraphConnection	Create an Activity Graph connection for the WebCenter Portal's Personalization service.	Online
deleteWCPSPeopleConnection	Create a People connection for the WebCenter Portal's Personalization service.	Online
deleteWCPSCustomConnection	Create a custom connection for the WebCenter Portal's Personalization service.	Online

10.13.1 createWCPSCMISConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.1.1 Description

Creates a CMIS (Content Management Interoperability Service) connection for the Personalization service.

10.13.1.2 Syntax

```
createWCPSCMISConnection(name, repositoryId, host, port, [scheme, namespace,
isDefault, path, pathPrepend, servletPathPart, rewriteUrls, pathTrim,
timeoutInMillisecs, propagateTimeoutExceptions, server])
```

Argument	Definition
name	Connection name. The name must be unique for this connection type within a namespace.
repositoryId	CMIS repository ID. Typically, the name of the Oracle WebCenter Content repository connection.
host	Hostname of the server hosting the CMIS REST service. Typically, the machine name of the WC_Spaces managed server.
port	Port of the server hosting the CMIS REST service. Typically, the port number of the WC_Spaces managed server.
scheme	Optional. HTTP scheme for accessing the CMIS REST service. Valid options are http and https. Defaults to http.
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection. If not specified or set to none, the connection is available to all namespaces.
isDefault	Optional. Indicates whether this connection is the default CMIS connection. Valid values are 1 (true) or 0 (false). Defaults to 0.

Argument	Definition
path	Optional. CMIS service URL path. Defaults to /api/cmis/repository/<repositoryId>.
pathPrepend	Optional. Base CMIS service URL path to prepend to the servletPathPart and path. Defaults to /rest.
servletPathPart	Optional. Servlet section of the CMIS service URL path.
rewriteUrls	Optional. Specifies how to rewrite URLs returned from the CMIS REST service. Valid options are producer, consumer, and none. Defaults to none. For more details, see 'Managing Personalization in WebCenter Portal' in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
pathTrim	Optional. Specifies the path parts to trim from URLs returned from the CMIS REST service. Defaults to None. For more details, see 'Managing Personalization in WebCenter Portal' in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
timeoutInMillisecs	Optional. Timeout in milliseconds (as a string) to wait for CMIS calls to return, or None for no timeout. Defaults to None.
propagateTimeoutExceptions	Optional. Valid values are 1 (true) and 0 (false). When set to 1, CMIS call timeouts raise an exception. When set to 0, exceptions are not raised.
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the WC_Utilities server hosts Personalization services.

10.13.1.3 Example

The following example creates a CMIS connection:

```
wls:/weblogic/serverConfig>createWCPSCMISConnection(name='Repos1CMISConnection',
repositoryId='ucm11g-server', host='myhost.com', port=8888, scheme='http',
isDefault=1)
```

10.13.2 createWCPSActivityGraphConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.2.1 Description

Creates an Activity Graph connection for the Personalization service.

10.13.2.2 Syntax

```
createWCPSActivityGraphConnection(name, host, port, [scheme], [namespace],
  [isDefault], [restResourceIndex], [rewriteUrls], [pathTrim], [server])
```

Argument	Definition
name	Connection name. Must be unique for this connection type within a namespace.

Argument	Definition
`host`	Hostname of the server hosting the Activity Graph REST service. Typically, the machine name of the `WC_Spaces` managed server.
`port`	Port of the server hosting the Activity Graph service. Typically, the port number of the `WC_Spaces` managed server.
`scheme`	Optional. HTTP scheme for accessing the Activity Graph service. Valid options are `http` and `https`. Defaults to `http`.
`namespace`	Optional. WebCenter Portal's Personalization connection namespace for the connection. If not specified or set to `none`, the connection is available to all namespaces.
`isDefault`	Optional. Indicates whether this connection is the default Activity Graph connection. Valid values are `1 (true)` or `0 (false)`. Defaults to `0`.
`restResourceIndex`	Optional. URL path for the resourceIndex of the REST server. Defaults to `/rest/api/resourceIndex`.
`rewriteUrls`	Optional. Specifies how to rewrite URLs returned from the Activity Graph REST service. Valid options are `producer`, `consumer`, and `none`. Defaults to `none`. For more details, see 'Managing Personalization in WebCenter Portal' in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
`pathTrim`	Optional. Specifies the path parts to trim from URLs returned from the Activity Graph REST service. Defaults to `None`. For more details, see 'Managing Personalization in WebCenter Portal' in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
`server`	Optional. Name of the Managed Server hosting the WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.

10.13.2.3 Example

The following example creates an Activity Graph connection in a particular namespace:

```
wls:/weblogic/serverConfig> createWCPSActivityGraphConnection(name='AGConnection',
host='myhost.com', port=8888, namespace='myNamespace')
```

10.13.3 createWCPSPeopleConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.3.1 Description

Creates a People connection for the Personalization service.

10.13.3.2 Syntax

```
createWCPSPeopleConnection(name, host, port, [scheme], [namespace],
  [isDefault], [restResourceIndex], [rewriteUrls], [pathTrim], [server])
```

Argument	Definition
name	Connection name. Must be unique for this connection type within a namespace.
host	Hostname of the server hosting the People Connection REST service. Typically, the machine name of the WC_Spaces managed server.
port	Port of the server hosting the People Connection service. Typically, the port number of the WC_Spaces managed server.
scheme	Optional. HTTP scheme for accessing the People Connection service. Valid options are http and https. Defaults to http.
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection. If not specified or set to none, the connection is available to all namespaces.
isDefault	Optional. Indicates whether this connection is the default People connection. Valid values are 1 (true) or 0 (false). Defaults to 0.
restResourceIndex	Optional. URL path for the resourceIndex of the REST server. Defaults to /rest/api/resourceIndex.
rewriteUrls	Optional. Specifies how to rewrite URLs returned from the People Connection REST service. Valid options are producer, consumer, and none. Defaults to none. For more details, see 'Managing Personalization in WebCenter Portal' in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
pathTrim	Optional. Specifies the path parts to trim from URLs returned from the People Connection service. Defaults to None. For more details, see 'Managing Personalization in WebCenter Portal' in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the WC_Utilities server hosts Personalization services.

10.13.3.3 Example

The following example creates a People connection in the default namespace:

```
wls:/weblogic/serverConfig> createWCPSPeopleConnection(name='PeopleConnection',
host='myhost.com', port=8888)
```

10.13.4 createWCPSCustomConnection

Use with WLST: Online

10.13.4.1 Description

Creates a connection of a specific type for the Personalization service.

Custom connection types are used with custom data providers and property locators.

10.13.4.2 Syntax

```
createWCPSCustomConnection(name, type, [namespace], [properties], [server])
```

Argument	Definition
name	Connection name. Must be unique for this connection type within a namespace.
type	Custom connection type specific to the custom data provider or property locator implementation.
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection. If not specified or set to `none`, the connection is available to all namespaces.
`properties`	Optional. Dictionary of connection properties and values. The set of properties is specific to the connection type. All values in the dictionary must be strings.
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.

10.13.4.3 Example

The following example creates an Activity Graph connection in a particular namespace:

```
wls:/weblogic/serverConfig> createWCPSCustomConnection(name='CustomConnection',
type='my.connection.type', properties={ 'prop1': 'value1', 'prop2', value2' })
```

10.13.5 listWCPSCMISConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.5.1 Description

Lists all CMIS (Content Management Interoperability Service) connections configured for the Personalization service or lists a single connection.

10.13.5.2 Syntax

```
listWCPSCMISConnections([server], [verbose], [name], [namespace])
```

Argument	Definition
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.
verbose	Optional. Controls verbose or concise output. Valid options are `1` and `0`. When set to `1`, this command lists the CMIS connections and their properties. When set to `0`, this command lists connection names only. Defaults to `1`.
name	Optional. Name of an existing connection. If not specified or set to None, then all connections are listed.

Argument	Definition
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection you want to list. If not specified or set to `none`, this command lists connections configured to be available in all namespaces.

10.13.5.3 Example

The following example lists the names of all the CMIS connections:

```
wls:/weblogic/serverConfig> listWCPSCMISConnections(verbose=0)

Repos1CMISConnection
Repos2CMISConnection
```

The following example lists the details of one CMIS connection:

```
wls:/weblogic/serverConfig> listWCPSCMISConnections(name='Repos1CMISConnection')

----------------------
Repos1CMISConnection (type=cmis.provider.connection, namespace=*)
----------------------
host: myhost.com
isDefault: false
path: /api/cmis/repository/repo1
pathPrepend: /rest
port: 8888
repositoryId: ucm11g-server
rewriteUrls: none
scheme: http
```

10.13.6 listWCPSActivityGraphConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.6.1 Description

Lists all Activity Graph connections configured for the Personalization service or lists a single connection.

10.13.6.2 Syntax

```
listWCPSActivityGraphConnections([server], [verbose], [name], [namespace])
```

Argument	Definition
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.
verbose	Optional. Controls verbose or concise output. Valid options are `1` and `0`. When set to `1`, this command lists the Activity Graph connections and their properties. When set to `0`, this command lists connection names only. Defaults to `1`.
name	Optional. Name of an existing connection. If not specified or set to None, then all connections are listed.

Argument	Definition
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection you want to list. If not specified or set to `none`, this command lists connections configured to be available in all namespaces.

10.13.6.3 Example

The following example lists the names of all the Activity Graph connections:

```
wls:/weblogic/serverConfig> listWCPSActivityGraphConnections(verbose=0)

AG1Connection
AG2Connection
```

The following example lists the details of one Activity Graph connection:

```
wls:/weblogic/serverConfig> listWCPSActivityGraphConnections(name='AG1Connection')

----------------------
AG1Connection (type=activity.provider.connection, namespace=*)
----------------------
host: myhost.com
isDefault: false
port: 8888
restResourceIndex: /rest/api/resourceIndex
rewriteUrls: producer
scheme: http
```

10.13.7 listWCPSPeopleConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.7.1 Description

Lists all People connections configured for the Personalization service or lists a single connection.

10.13.7.2 Syntax

```
listWCPSPeopleConnections([server], [verbose], [name], [namespace])
```

Argument	Definition
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.
verbose	Optional. Controls verbose or concise output. Valid options are `1` and `0`. When set to `1`, this command lists the People connections and their properties. When set to `0`, this command lists connection names only. Defaults to `1`.
name	Optional. Name of an existing connection. If not specified or set to None, then all connections are listed.

Argument	Definition
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection you want to list. If not specified or set to `none`, this command lists connections configured to be available in all namespaces.

10.13.7.3 Example

The following example lists the names of all the People connections:

```
wls:/weblogic/serverConfig> listWCPSPeopleConnections(verbose=0)

People1Connection
People2Connection
```

The following example lists the details of one People connection:

```
wls:/weblogic/serverConfig> listWCPSPeopleConnections(name='PeopleConnection')

----------------------
PeopleConnection (type=people.service.connection, namespace=*)
----------------------
host: myhost.com
isDefault: false
port: 8888
restResourceIndex: /rest/api/resourceIndex
rewriteUrls: producer
scheme: http
```

10.13.8 listWCPSCustomConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.8.1 Description

Lists all connections of a particular type configured for the Personalization service or lists a single connection.

Custom connection types are used with custom data providers and property locators.

10.13.8.2 Syntax

```
listWCPSCustomConnections(type, [server], [verbose], [name], [namespace])
```

Argument	Definition
type	Custom connection type specific to the custom data provider or property locator implementation.
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.
verbose	Optional. Controls verbose or concise output. Valid options are `1` and `0`. When set to `1`, this command lists the connections and their properties. When set to `0`, this command lists connection names only. Defaults to `1`.

Argument	Definition
name	Optional. Name of an existing connection. If not specified or set to None, then all connections are listed.
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection you want to list. If not specified or set to `none`, this command lists connections configured to be available in all namespaces.

10.13.8.3 Example

The following example lists the names of all connections with the type `my.connection.type`:

```
wls:/weblogic/serverConfig> listWCPSCustomConnections(type='my.connection.type',
verbose=0)

Custom1Connection
Custom2Connection
```

The following example lists the details of one custom connection:

```
wls:/weblogic/serverConfig> listWCPSPeopleConnections(type='my.connection.type',
name='CustomConnection')

----------------------
CustomConnection (type=my.connection.type, namespace=*)
----------------------
host: myhost.com
isDefault: false
port: 8888
customConnectionProperty: someValue
scheme: http
```

10.13.9 deleteWCPSCMISConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.9.1 Description

Deletes a CMIS (Content Management Interoperability Service) connection configured for the Personalization service.

10.13.9.2 Syntax

```
deleteWCPSCMISConnection(name, [namespace, server])
```

Argument	Definition
name	Connection name.
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection you want to delete. If not specified or set to `none`, this command deletes connections configured to be available in all namespaces.

Argument	Definition
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.

10.13.9.3 Example

The following example deletes a CMIS connection:

```
wls:/weblogic/serverConfig>deleteWCPSCMISConnection(name='ReposCMISConnection')
```

10.13.10 deleteWCPSActivityGraphConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.10.1 Description

Deletes an Activity Graph connection configured for the Personalization service.

10.13.10.2 Syntax

```
deleteWCPSActivityGraphConnection(name, [namespace, server])
```

Argument	Definition
name	Connection name.
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection you want to delete. If not specified or set to `none`, this command deletes connections configured to be available in all namespaces.
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.

10.13.10.3 Example

The following example deletes an Activity Graph connection:

```
wls:/weblogic/serverConfig>deleteWCPSActivityGraphConnection(name='AGConnection')
```

10.13.11 deleteWCPSPeopleConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.11.1 Description

Deletes a People connection configured for the Personalization service.

10.13.11.2 Syntax

```
deleteWCPSPeopleConnection(name, [namespace, server])
```

Argument	Definition
name	Connection name.
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection you want to delete. If not specified or set to `none`, this command deletes connections configured to be available in all namespaces.
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.

10.13.11.3 Example

The following example deletes a People connection:

```
wls:/weblogic/serverConfig>deleteWCPSPeopleConnection(name='PeopleConnection')
```

10.13.12 deleteWCPSCustomConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.12.1 Description

Deletes a custom connection configured for the Portal's Personalization service.

10.13.12.2 Syntax

```
deleteWCPSCustomConnection(name, type, [namespace, server])
```

Argument	Definition
name	Connection name.
type	Custom connection type.
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection you want to delete. If not specified or set to `none`, this command deletes connections configured to be available in all namespaces.
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.

10.13.12.3 Example

The following example deletes a custom connection:

```
wls:/weblogic/serverConfig>deleteWCPSCustomConnection(name='CustomConnection',
type='my.connection.type')
```

10.13.13 setWCPSConnectionProperty

Module: Oracle WebCenter Portal

Use with WLST: Online

10.13.13.1 Description

Add, modify, or delete properties of an existing connection for the Personalization service. The properties supported by a connection are specific to the connection type:

- CMIS connections support the following properties: `repositoryId, host, port, scheme, path, pathPrepend, servletPathPart, rewriteUrls, pathTrim, isDefault, timeoutInMillisecs, propagateTimeoutException`

  See also, createWCPSCMISConnection.

- Activity Graph and People Connections support the following properties: `host, port, scheme, restResourceIndex, rewriteUrls, pathTrim, isDefault`

  See also, createWCPSActivityGraphConnection and createWCPSPeopleConnection.

10.13.13.2 Syntax

```
setWCPSConnectionProperty(connectionName, connectionType, propertyName,
propertyValue, [namespace], [server])
```

Argument	Definition
connectionName	Connection name.
connectionType	Connection type. Valid values are `WCPS_CMIS_CONNECTION_TYPE`, `WCPS_AG_CONNECTION_TYPE`, and `WCPS_PC_CONNECTION_TYPE` for CMIS, Activity Graph, and People Connections, respectively. Alternatively, any valid, custom connection type can be specified
propertyName	Property name.
propertyValue	Property value as a string. Use `None` to remove a property value from the connection.
namespace	Optional. WebCenter Portal's Personalization connection namespace for the connection you want to change. If not specified or set to `none`, this command modifies properties of connections configured to be available in all namespaces.
server	Optional. Name of the Managed Server hosting WebCenter Portal's Personalization service. This parameter is only required in a nondefault deployment configuration. No value is required for a default deployment where the `WC_Utilities` server hosts Personalization services.

10.13.13.3 Example

The following example changes or adds a property to a CMIS connection:

```
wls:/weblogic/serverConfig>
setWCPSConnectionProperty(connectionName='ReposCMISConnection',

connectionType=WCPS_CMIS_CONNECTION_TYPE,

propertyName='propagateTimeoutExceptions', propertyValue=0)
```

The following example removes a property from a custom connection.:

```
wls:/weblogic/serverConfig>
setWCPSConnectionProperty(connectionName='CustomConnection',
connectionType='my.connection.type', propertyName='prop2', propertyValue=None)
```

10.14 Portlet Producers

Use the commands listed in Table 10–21 to manage portlet producers used in WebCenter Portal applications.

All configuration changes made using these WebCenter Portal WLST commands are immediately available in the WebCenter Portal application.

Table 10–21 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
listPDKJavaProducers	List registered Oracle PDK-Java producers.	Online
deregisterPDKJavaProducer	Deregister an Oracle PDK-Java producer, deleting the associated connection.	Online
registerPageletProducer	Create and register a pagelet producer.	Online
setPageletProducer	Edit pagelet producer registration details.	Online
listPageletProducers	List pagelet producer registration details.	Online
deregisterPageletProducer	Deregister a pagelet producer, deleting the associated connection.	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 Portal.	Online
deregisterOOTBProducers	Deregister out-of-the-box producers provided with Oracle WebCenter Portal.	Online
registerSampleProducers	Register the sample producers provided with Oracle WebCenter Portal.	Online
deregisterSampleProducers	Deregister sample producers.	Online

10.14.1 registerWSRPProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.1.1 Description

Creates a connection to a WSRP portlet producer and registers the WRSP producer with a named WebCenter Portal 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.

10.14.1.2 Syntax

```
registerWSRPProducer(appName, name, url, [proxyHost], [proxyPort],
[timeout],[externalApp],[registrationProperties],[tokenType],[issuer],[defUser],
[keyStorePath],[keyStorePswd],[sigKeyAlias],[sigKeyPswd],[encKeyAlias],
[encKeyPswd],[recptAlias], [enforcePolicyURI], [server],[applicationVersion])
```

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

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 Portal 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 10.14.5, "listWSRPProducerRegistrationProperties".

Argument	Definition
tokenType	Optional. Type of token profile to use for authentication with this WSRP producer. When the argument `enforcePolicyURI=1`, valid values are: ■ **USERNAME_WITHOUT_PASSWORD** (oracle/wss10_username_id_propagation_with_msg_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Credentials (*user name* only) are included in outbound SOAP request messages through a WS-Security UsernameToken header. No password is included. Message protection is provided using WS-Security 1.0's Basic128 suite of asymmetric key technologies. Specifically, RSA key mechanisms for message confidentiality, SHA-1 hashing algorithm for message integrity, and AES-128 bit encryption. ■ **USERNAME_WITH_PASSWORD** (oracle/wss10_username_token_with_message_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security v1.0 standard. Both plain text and digest mechanisms are supported. This policy 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. 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)—This policy provides message-level integrity protection and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation. This policy uses WS-Security's Basic 128 suite of asymmetric key technologies and SHA-1 hashing algorithm for message integrity. When this policy is selected, the recipient key alias (recptAlias) must be disabled. ■ **SAML_TOKEN_WITH_MSG_PROTECTION** (oracle/wss10_saml_token_with_message_protection_client_policy)—This policy provides message-level protection (integrity and confidentiality) and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches. This policy 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. and SHA-1 hashing algorithm for message integrity.

Argument	Definition
tokenType continued...	■ **WSS11_SAML_TOKEN_WITH_MSG_PROTECTION** (oracle/wss11_saml_token_with_message_protection_client_policy)—This policy provides message-level protection (integrity and confidentiality) and SAML token population for outbound SOAP requests in accordance with the WS-Security 1.1 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation. This policy uses the symmetric key technology for signing and encryption, and WS-Security's Basic 128 suite of asymmetric key technologies for endorsing signatures. ■ **WSS10_SAML_TOKEN_ONLY** (oracle/wss10_saml_token_client_policy)—This policy provides SAML-based authentication for outbound SOAP request messages in accordance with the WS-Security 1.0 standard. The policy propagates user identity and is typically used in intra departmental deployments where message protection and integrity checks are not required. This policy does not require any keystore configuration. If the argument `enforcePolicyURI=0`, you can specify any *valid* Oracle Web Services Manager (OWSM) policy URI for the `tokenType` argument.
issuer	Optional. Name of the issuer of the token. The issuer name is the entity that vouches for the verification of the subject. For example: `www.oracle.com`. This argument only applies when the tokenType is: `SAML_TOKEN_WITH_MSG_PROTECTION`, `SAML_TOKEN_WITH_MSG_INTEGRITY`, `WSS10_SAML_TOKEN_ONLY`, `WSS11_SAML_TOKEN_WITH_MSG_PROTECTION`.
defUser	Optional. User name to assert to the remote producer when the user is not authenticated with the WebCenter Portal application. When unauthenticated, the identity *anonymous* is associated with the application user. The value *anonymous* may be inappropriate for the remote producer, so you may need to specify an alternative identity here. Keep in mind though, that in this case, the WebCenter Portal application has not authenticated the user so the default user you specify should be a low privileged user in the remote producer. If the user has authenticated to the application, the user's identity is asserted rather than the default user. This argument only applies when the tokenType is: `USERNAME_WITHOUT_PASSWORD`, `SAML_TOKEN_WITH_MSG_PROTECTION`, `SAML_TOKEN_WITH_MSG_INTEGRITY`, `WSS10_SAML_TOKEN_ONLY`, `WSS11_SAML_TOKEN_WITH_MSG_PROTECTION`.
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.
sigKeyPswd	Optional. Password for accessing the key identified by the alias that is specified using the `sigKeyAlias` argument.

Argument	Definition
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`.
enforcePolicyURI	Optional. Valid values are `1` (true) and `0` (false). When set to 1, users must specify one of the following token profiles for the `tokenType` argument: `USERNAME_WITHOUT_PASSWORD`, `USERNAME_WITH_PASSWORD`, `SAML_TOKEN_WITH_MSG_PROTECTION`, `SAML_TOKEN_WITH_MSG_INTEGRITY`, `WSS11_SAML_TOKEN_WITH_MSG_PROTECTION`,`WSS10_SAML_TOKEN_ONLY` When set to `0`, users can specify any Oracle Web Services Manager (OWSM) policy URI. The user must ensure that the OWSM policy specified is valid. The default value is `1`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.1.3 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='WSS11_SAML_TOKEN_WITH_MSG_PROTECTION', issuer='www.oracle.com',
defUser='anonymous', keyStorePath='/keys/mykeystore.jks', keyStorePswd='Test1',
sigKeyAlias='mysigalias',sigKeyPswd='mysigpswd', encKeyAlias='myencalias',
encKeyPswd='myencpswd', recptAlias='myrcptalias')
```

10.14.2 setWSRPProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.2.1 Description

Edits registration details for an existing WSRP producer.

10.14.2.2 Syntax

```
setWSRPProducer(appName, name, [url], [proxyHost], [proxyPort], [timeout],
[externalApp], [tokenType],[issuer], [defUser], [keyStorePath], [keyStorePswd]
[sigKeyAlias], [sigKeyPswd], [encKeyAlias], [encKeyPswd], [recptAlias],
[enforcePolicyURI], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal 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 Portal 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. When the argument `enforcePolicyURI=1`, valid values are: ■ **USERNAME_WITHOUT_PASSWORD** (oracle/wss10_username_id_propagation_with_msg_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and identity propagation for outbound SOAP requests in accordance with the WS-Security 1.0 standard. Credentials (*user name* only) are included in outbound SOAP request messages through a WS-Security UsernameToken header. No password is included. Message protection is provided using WS-Security 1.0'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. ■ **USERNAME_WITH_PASSWORD** (oracle/wss10_username_token_with_message_protection_client_policy)—This policy provides message protection (integrity and confidentiality) and authentication for outbound SOAP requests in accordance with the WS-Security v1.0 standard. Both plain text and digest mechanisms are supported. This policy 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. 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)—This policy provides message-level integrity and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation. This policy uses WS-Security's Basic 128 suite of asymmetric key technologies and SHA-1 hashing algorithm for message integrity. When this policy is selected, the recipient key alias (recptAlias) must be disabled. ■ **SAML_TOKEN_WITH_MSG_PROTECTION** (oracle/wss10_saml_token_with_message_protection_client_policy)—This policy provides message-level protection (integrity and confidentiality) and SAML-based authentication for outbound SOAP requests in accordance with the WS-Security 1.0 standard. The Web service consumer includes a SAML token in the SOAP header and the confirmation type is sender-vouches. This policy 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. and SHA-1 hashing algorithm for message integrity.

Argument	Definition
tokenType continued...	■ **WSS11_SAML_TOKEN_WITH_MSG_PROTECTION** (oracle/wss11_saml_token_with_message_protection_client_policy)—This policy enables message-level protection (integrity and confidentiality) and SAML token population for outbound SOAP requests in accordance with the WS-Security 1.1 standard. A SAML token, included in the SOAP message, is used in SAML-based authentication with sender vouches confirmation. This policy uses the symmetric key technology for signing and encryption, and WS-Security's Basic 128 suite of asymmetric key technologies for endorsing signatures. ■ **WSS10_SAML_TOKEN_ONLY** (oracle/wss10_saml_token_client_policy)—This policy includes SAML-tokens in outbound SOAP request messages in accordance with the WS-Security 1.0 standard. The policy propagates user identity and is typically used in intra departmental deployments where message protection and integrity checks are not required. This policy does not require any keystore configuration. If the argument `enforcePolicyURI=0`, you can specify any *valid* Oracle Web Services Manager (OWSM) policy URI for the `tokenType` argument.
issuer	Optional. Name of the issuer of the token. The issuer name is the entity that vouches for the verification of the subject. For example: `www.oracle.com`. This argument only applies when the tokenType is: `SAML_TOKEN_WITH_MSG_PROTECTION`, `SAML_TOKEN_WITH_MSG_INTEGRITY`, `WSS10_SAML_TOKEN_ONLY`, `WSS11_SAML_TOKEN_WITH_MSG_PROTECTION`.
defUser	Optional. User name to assert to the remote producer when the user is not authenticated with the WebCenter Portal application. When unauthenticated, the identity *anonymous* is associated with the application user. The value *anonymous* may be inappropriate for the remote producer, so you may need to specify an alternative identity here. Keep in mind though, that in this case, the WebCenter Portal application has not authenticated the user so the default user you specify should be a low privileged user in the remote producer. If the user has authenticated to the application, the user's identity is asserted rather than the default user. This argument only applies when the tokenType is: `USERNAME_WITHOUT_PASSWORD`, `SAML_TOKEN_WITH_MSG_PROTECTION`, `SAML_TOKEN_WITH_MSG_INTEGRITY`, `WSS10_SAML_TOKEN_ONLY`, `WSS11_SAML_TOKEN_WITH_MSG_PROTECTION`.
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.
sigKeyPswd	Optional. Password for accessing the key identified by the alias that is specified using the `sigKeyAlias` argument.

Argument	Definition
encKeyAlias	Optional. Key alias used by the producer to encrypt the return message. A valid value is one of the key aliases that is located in the specified key store. If not specified, the producer uses the signing key for encrypting the return message.
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 outbound message to the producer. Do not specify a recipient key alias when the `tokenType` is `SAML_TOKEN_WITH_MSG_INTEGRITY`.
enforcePolicyURI	Optional. Valid values are `1` (true) and `0` (false). When set to 1, users must specify one of the following token profiles for the `tokenType` argument: `USERNAME_WITHOUT_PASSWORD`, `USERNAME_WITH_PASSWORD`, `SAML_TOKEN_WITH_MSG_PROTECTION`, `SAML_TOKEN_WITH_MSG_INTEGRITY`, `WSS11_SAML_TOKEN_WITH_MSG_PROTECTION`, `WSS10_SAML_TOKEN_ONLY` When set to `0`, users can specify any Oracle Web Services Manager (OWSM) policy URI. The user must ensure that the OWSM policy specified is valid. The default value is `1`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.2.3 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='WSS11_SAML_TOKEN_WITH_MSG_PROTECTION',
issuer='www.oracle.com',  defUser='anonymous',
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='')
```

10.14.3 listWSRPProducers

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.3.1 Description

Lists WSRP producer registration details.

10.14.3.2 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 Portal application are listed.
verbose	Optional. Displays WSRP producer connection details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listWSRPProducers` lists all connection properties. When set to `0`, `listWSRPProducers` lists connection names only. This argument defaults to `1`. If you set this argument to `0`, do not specify the `names` argument.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.3.3 Example

The following example lists all the WSRP producers registered with an application named `myApp`:

```
wls:/weblogic/serverConfig> listWSRPProducers(appName='myApp', verbose=0)


----------------------
WSRPSamples-connection
----------------------
```

The following example lists detailed connection information for a WSRP producer registered as `WSRPSamples-connection` with an application named `myApp`:

```
wls:/weblogic/serverConfig> listWSRPProducers(appName='myApp',
name='WSRPSamples-connection', verbose=1)


----------------------
WSRPSamples-connection
----------------------
Connection Name: WSRPSamples-connection
Web Service Connection Name: WSRPSamples-connection-wsconn
Proxy Host: None
Proxy Port: None
Timeout: 0
WSDL URL: http://pspencer-lnx.uk.oracle.com:7777/portletapp/portlets/wsrp2?WSDL
```

10.14.4 deregisterWSRPProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.4.1 Description

Deregisters a WSRP producer, and deletes the associated WSRP and Web Service connections.

10.14.4.2 Syntax

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

Argument	Definition
appName	Name of the WebCenter Portal application where the producer is registered.
name	Name of an existing WSRP producer.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.4.3 Example

The following example deregisters the `WSRPSamples` producer in an application named `webcenter`.

```
wls:/weblogic/serverConfig> deregisterWSRPProducer(appName='webcenter',
name='WSRPSamples')
```

10.14.5 listWSRPProducerRegistrationProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.5.1 Description

Lists registration properties supported by a WSRP portlet producer.

10.14.5.2 Syntax

```
listWSRPProducerRegistrationProperties(appName, url,[proxyHost, [proxyPort],
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

Argument	Definition

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 Portal 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.5.3 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
```

10.14.6 listWSRPProducerUserCategories

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.6.1 Description

Lists any user categories that a WSRP producer might support. WebCenter Portal users can use the WLST command mapWSRPProducerUserCategory to map application roles to a producer's user category.

10.14.6.2 Syntax

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

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing WSRP producer.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.6.3 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
```

10.14.7 mapWSRPProducerUserCategory

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.7.1 Description

Maps a role that is defined in the specified WebCenter Portal application to a user category supported by a WSRP producer. The user categories may be found using listWSRPProducerUserCategories.

10.14.7.2 Syntax

```
mapWSRPProducerUserCategory(appName, name, localRole, producerUserCategory,
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing WSRP producer.
localRole	Name of the WebCenter Portal application role to be mapped.
producerUserCategory	WSRP producer user category to which the WebCenter Portal role will be mapped.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.7.3 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')
```

10.14.8 registerPDKJavaProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.8.1 Description

Creates a connection to an Oracle PDK-Java portlet producer and registers the Oracle PDK-Java producer with a named WebCenter Portal application.

10.14.8.2 Syntax

```
registerPDKJavaProducer(appName, name, url, [serviceId], [proxyHost,
[proxyPort]], [subscriberId], [sharedKey], [timeout],
[establishSession],[externalApp], [mapUser], [server], [applicationVersion])
```

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

Argument	Definition

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 Portal 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 Portal application in this instance) passes the value for subscriberId to the producer. The producer may be coded to use the subscriber ID.
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 Portal 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 `1` (true) and `0` (false). The default for this argument is `0`. When sessions are enabled (`1`), 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 `0`.
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 user name from the external application should be passed to the producer. Valid values are `1` (true) and `0` (false). This argument defaults to `1`.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.8.3 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')
```

10.14.9 setPDKJavaProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.9.1 Description

Edits registration details for an existing PDK-Java producer.

10.14.9.2 Syntax

```
setPDKJavaProducer(appName, name, url, [serviceId], [proxyHost, [proxyPort]],
[subscriberId], [sharedKey], [timeout], [establishSession], [externalApp],
[mapUser], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
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`

Argument	Definition

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 Portal 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 Portal 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 Portal pages. 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 `1` (true) and `0` (false). You should enable sessions using the `establishSession` argument, as well as the `sharedKey` argument. When sessions are enabled (`1`), 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 `0`.
externalApp	Optional. Name of the external application associated with this producer.
mapUser	Optional. Flag indicating whether the mapped user name from the external application should be passed to the producer. Valid values are `1` (true) and `0` (false).

Argument	Definition
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.14.9.3 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)
```

10.14.10 deregisterPDKJavaProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.10.1 Description

Deregisters an Oracle PDK-Java producer and deletes the associated connection, for a named WebCenter Portal application.

10.14.10.2 Syntax

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

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing PDK-Java producer.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.14.10.3 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
```

10.14.11 listPDKJavaProducers

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.11.1 Description

Lists details for one or more Oracle PDK-Java producers registered with a named WebCenter Portal application.

10.14.11.2 Syntax

```
listPDKJavaProducers(appName, [name],[verbose], [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application are listed.
verbose	Optional. Displays PDK-Java producer connection details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listPDKJavaProducers` lists all connection properties. When set to `0`, `listPDKJavaProducers` lists connection names only. This argument defaults to `1`. If you set this argument to `0`, do not specify the `name` argument.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.11.3 Example

The following example lists all the connection properties (verbose mode) for the JPDKSamples producer.

```
wls:/weblogic/serverConfig> listPDKJavaProducers(appName='webcenter',
name='JPDKSamples', verbose=1)
----------------------
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
```

10.14.12 registerPageletProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.12.1 Description

Registers a pagelet producer with a named WebCenter Portal application.

10.14.12.2 Syntax

```
registerPageletProducer(appName, name, url, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application for which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal application. The name you specify here appears in Composer under the *Mash-ups >Pagelet Producers* folder (by default).

url	URL required to access WebCenter Portal's Pagelet Producer. Use the syntax:
	protocol://*host.domain*:*port_number*/pagelets
	The URL must include a fully-qualified domain name. For example:
	`http://myhost.example.com:7778/pagelets`
	If pagelets carry secure data, the URL registered must use the `https` protocol. For example:
	`https://myhost.com:7779/pagelets`
	Note: In the Spaces application, if the Pagelet Producer URL is protected by Oracle Access Manager (OAM), the URL to the pagelet catalog must be excluded (mapped directly without access control), or the catalog will appear to be empty when using REST. The pagelet catalog URL is: `http://<proxy_host>:<proxy_port>/api/v2/ensemble/pagelets`

server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.12.3 Example

The following example registers a pagelet producer with an application named `webcenter`.

```
wls:/weblogic/serverConfig> registerPageletProducer(appName='webcenter',
name='MyPageletProducer', url='http://myhost.com:7001/pagelets')
```

10.14.13 setPageletProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.13.1 Description

Edits connection details for an existing pagelet producer.

10.14.13.2 Syntax

```
setPageletProducer(appName, name, [url, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing pagelet producer connection.
url	Optional. URL required to access WebCenter Portal's Pagelet Producer. Use the syntax: *protocol*://*host.domain*:*port_number*/pagelets The URL must include a fully-qualified domain name. For example: http://myhost.example.com:7778/pagelets **Note:** In the Spaces application, if the Pagelet Producer URL is protected by Oracle Access Manager (OAM), the URL to the pagelet catalog must be excluded (mapped directly without access control), or the catalog will appear to be empty when using REST. The pagelet catalog URL is: http://<proxy_host>:<proxy_port>/api/v2/ensemble/pagelets
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.14.13.3 Example

The following example updates administrator user name and password details for an existing pagelet producer named MyPageletProducer:

```
wls:/weblogic/serverConfig> setPageletProducer(appName='webcenter',
name='MyPageletProducer', url='http://mypagelethost.com:7778/pagelets')
```

10.14.14 listPageletProducers

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.14.1 Description

Lists connection details for one or all pagelet producers registered with a named WebCenter Portal application.

10.14.14.2 Syntax

```
listPageletProducers(appName, [name],[verbose], [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Optional. Name of an existing pagelet producer connection. Use this argument to view connection details for a specific pagelet producer. If omitted, connection details for all pagelet producers configured for this WebCenter Portal application are listed.
verbose	Optional. Displays pagelet producer connection details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listPageletProducers` lists all connection properties. When set to `0`, `listPageletProducers` lists connection names only. This argument defaults to `1`. If you set this argument to `0`, do not specify the `name` argument.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.14.3 Example

The following example lists connection names and details for all pagelet producers currently registered for a WebCenter Portal application name `MyWebCenterApp`:

```
wls:/weblogic/serverConfig> listPageletProducers(appName='MyWebCenterApp',
verbose=1)
----------------------
MyPageletProducer
----------------------
URL: http://myhost.com:7001/pagelets
----------------------
TestPageletProducer
----------------------
URL: http://testhost.com:7002/pagelets
----------------------
```

The following example displays details for a single pagelet producer connection named `MyPageletProducer`:

```
wls:/weblogic/serverConfig> listPageletProducers(appName='webcenter',
name='MyPageletProducer', verbose=1)
----------------------
MyPageletProducer
----------------------
URL: http://myhost.com:7001/pagelets
```

10.14.15 deregisterPageletProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.15.1 Description

Deregisters a pagelet producer currently registered with a named WebCenter Portal application.

10.14.15.2 Syntax

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

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing pagelet producer connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.15.3 Example

The following example deregisters a pagelet producer connection named `MyPageletProducer` currently configured for a WebCenter Portal application name `MyWebCenterApp`:

```
wls:/weblogic/serverConfig> deregisterPageletProducer(appName='MyWebCenterApp',
name='MyPageletProducer')
```

10.14.16 refreshProducer

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.16.1 Description

Refreshes the metadata stored for a named producer to reflect the portlets that are currently offered by that producer.

10.14.16.2 Syntax

```
refreshProducer(appName, producerName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which the producer is registered.
producerName	Name of an existing producer.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.16.3 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.
```

10.14.17 registerOOTBProducers

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.17.1 Description

Registers several out-of-the-box producers provided with Oracle WebCenter Portal: OmniPortlet, Web Clipping, and WSRP Tools.

10.14.17.2 Syntax

```
registerOOTBProducers(producerHost, producerPort, appName, [server,
applicationVersion])
```

Argument	Definition
producerHost	Host name or IP address of the server hosting out-of-the-box producers. In a cluster fronted by a load balancer, enter the host name of the load balancer.
producerPort	Port number for the server hosting out-of-the-box producers. In a cluster, fronted by a load balancer, enter the port number of the load balancer.
appName	Name of the WebCenter Portal application in which the out-of-the-box producers are to be registered.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.17.3 Example

The following example registers out-of-the-box producers in a WebCenter Portal 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 WSRP Tools
Created connection wc-WSRPTools-wsconn
Created connection wc-WSRPTools
Producer connection wc-WSRPTools has been registered.
```

10.14.18 deregisterOOTBProducers

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.18.1 Description

Deregisters out-of-the-box producers provided with Oracle WebCenter Portal: OmniPortlet, Web Clipping, and WSRP Tools.

10.14.18.2 Syntax

```
deregisterOOTBProducers(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which the out-of-the-box producers are currently registered.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.18.3 Example

The following example deregisters out-of-the-box WebCenter Portal 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 WSRP Tools
Producer wc-WSRPTools has been deregistered.
wc-WSRPTools successfully deleted
```

```
wc-WSRPTools-wsconn successfully deleted
```

10.14.19 registerSampleProducers

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.19.1 Description

Registers the sample producers provided with Oracle WebCenter Portal with a named WebCenter Portal application. There are two sample producers — WSRP Samples and JPDK Samples.

10.14.19.2 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 Portal application in which the sample producers are to be registered.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.19.3 Example

The following example registers Oracle WebCenter Portal sample producers in an application named `myApp`.

```
wls:/weblogic/serverConfig> registerSampleProducers(producerHost='myhost.com',
producerPort=9999, appName='myApp')
```

10.14.20 deregisterSampleProducers

Module: Oracle WebCenter Portal

Use with WLST: Online

10.14.20.1 Description

Deregisters the Oracle WebCenter Portal sample producers (`WSRP Samples` and `JPDK Samples`) from a named WebCenter Portal application.

10.14.20.2 Syntax

```
deregisterSampleProducers(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.14.20.3 Example

The following example deregisters sample producers from a WebCenter Portal application named `myApp`.

```
wls:/weblogic/serverConfig> deregisterSampleProducers(appName='myApp')
```

10.15 RSS News Feeds

Use the commands listed in Table 10–22 to manage proxy settings for the RSS service.

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

Table 10–22 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
unsetRssProxyConfig	Delete proxy host and proxy port settings.	Online

10.15.1 getRssProxyConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.15.1.1 Description

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 your WebCenter Portal application.

10.15.1.2 Syntax

```
getRssProxyConfig(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.15.1.3 Example

The following example returns the proxy host and proxy port used by the RSS service in a WebCenter Portal application named `webcenter`.

```
wls:/weblogic/serverConfig> getRssProxyConfig(appName='webcenter')
```

10.15.2 setRssProxyConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.15.2.1 Description

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 your WebCenter Portal application.

10.15.2.2 Syntax

```
setRssProxyConfig(appName, proxyHost, proxyPort, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.15.2.3 Example

The following example sets the proxy host and proxy port used by the RSS service in a WebCenter Portal application named `webcenter`.

```
wls:/weblogic/serverConfig> setRssProxyConfig(appName='webcenter',
proxyHost='www-proxy.example.com', proxyPort='80')
```

10.15.3 unsetRssProxyConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.15.3.1 Description

Deletes the current proxy host and proxy port settings.

10.15.3.2 Syntax

```
unsetRssProxyConfig(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.15.3.3 Example

The following example deletes the proxy host and proxy port settings used by the RSS service in a WebCenter Portal application named `webcenter`.

```
wls:/weblogic/serverConfig> unsetRssProxyConfig(appName='webcenter')
```

10.16 Search - Oracle SES Search

Use the commands listed in Table 10–23 to manage Oracle Secure Enterprise Search (SES) connections and other Oracle SES search related properties for WebCenter Portal applications.

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

Table 10–23 Search - Oracle SES WLST Commands

Use this command...	To...	Use with WLST...
createSESConnection	Create a connection to an Oracle SES instance for a WebCenter Portal application.	Online
setSESConnection	Edit an existing Oracle SES search connection.	Online
listSESConnections	List individual or all Oracle SES search connections that are configured for a specific WebCenter Portal application.	Online
setSearchSESConfig	Configure search settings for an existing Oracle SES search connection.	Online

Table 10–23 (Cont.) Search - Oracle SES WLST Commands

Use this command...	To...	Use with WLST...
listSearchSESConfig	List Oracle SES properties for a WebCenter Portal application.	Online
createFederationTrustedEntity	Create a federation trusted entity on an Oracle (SES) instance.	Online

10.16.1 createSESConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.16.1.1 Description

Creates a connection to an Oracle Secure Enterprise Search (SES) instance for a WebCenter Portal application.

10.16.1.2 Syntax

```
createSESConnection(appName, name, url, appUser, appPassword,[default],
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal 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 Portal application uses to authenticate itself as a trusted application to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter Portal users. The specified user must be present in both the Oracle Identity Management server configured for the WebCenter Portal application and the Oracle Identity Management server configured for Oracle SES.
appPassword	Password for the user name specified.
default	Optional. Configures WebCenter Portal's Search service to actively use the search connection. Valid options are `1` (true) and `0` (false). Setting to `1` replaces any other search connection that is being used. Setting to `0` does not change the current Search service configuration. This argument defaults to `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.16.1.3 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 SES search connection for a WebCenter Portal application named `app1`.

```
wls:/weblogic/serverConfig> createSESConnection(appName='app1', name='SESConn1',
url='http://myhost.com:7777/search/query/OracleSearch', appUser='wpadmin',
appPassword='password', default=1)
```

10.16.2 setSESConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.16.2.1 Description

Edits an existing Oracle Secure Enterprise Search (SES) search connection.

10.16.2.2 Syntax

```
setSESConnection(appName, name, [url], [appUser],[appPassword],[default],
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application uses to log in to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter Portal users.
appPassword	Optional. Password that the WebCenter Portal application uses to log in to Oracle Secure Enterprise Search so that it may perform searches on behalf of WebCenter Portal users.
default	Optional. Configures WebCenter Portal's Search service to actively use the search connection. Valid options are `1` (true) and `0` (false). Setting to `1` replaces any other search connection that is being used. Setting to `0` does not change the current Search service configuration. This argument defaults to `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.16.2.3 Example

The following example modifies the URL of a search connection named `SESConn1` and makes the connection the active SES search connection for a WebCenter Portal application named `app1`.

```
wls:/weblogic/serverConfig> setSESConnection(appName='app1', name='SESConn1',
url='http://myhost.com:7777/search/query/OracleSearch', appUser='wpadmin',
appPassword='password', default=1)
```

10.16.3 listSESConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.16.3.1 Description

Lists the names of all Oracle Secure Enterprise Search (SES) search connections configured for a WebCenter Portal application.

10.16.3.2 Syntax

```
listSESConnections(appName, [verbose], [name], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application for which to perform this operation.
verbose	Optional. Displays search connection details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listSESConnections` lists all of the SES search connections that are configured for a WebCenter Portal application, along with their details. When set to `0`, `listSESConnections` lists connection names only. This argument defaults to `0`. If you set this argument to `0`, 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.16.3.3 Examples

The following example displays connection details for all SES search connections configured for a WebCenter Portal application named `WebCenterApp`.

```
wls:/weblogic/serverConfig> listSESConnections(appName='WebCenterApp', verbose=1)
```

The following example displays connection details for an SES search connection named `SESConn1`.

```
wls:/weblogic/serverConfig> listSESConnections(appName='WebCenterApp',
verbose=1, name='SESConn1')
```

10.16.4 setSearchSESConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.16.4.1 Description

Configures search settings for an existing Oracle Secure Enterprise Search (SES) search connection. If a parameter is not specified it is not modified.

10.16.4.2 Syntax

```
setSearchSESConfig(appName,[connectionName],[dataGroup],[topNRows], [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.16.4.3 Example

The following example specifies that the Search service must use the SES 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.
```

10.16.5 listSearchSESConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.16.5.1 Description

Lists SES search settings for a WebCenter Portal application.

10.16.5.2 Syntax

```
listSearchSESConfig(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application for which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.16.5.3 Example

The following example displays SES search configuration information for a WebCenter Portal application named `webcenter`.

```
wls:/weblogic/serverConfig> listSearchSESConfig(appName='webcenter')
Already in Domain Runtime Tree
-----------------
Search SES Config
-----------------
connectionName:  SESConn1
dataGroup:  group2
topNRows:  200
```

10.16.6 createFederationTrustedEntity

Module: Oracle WebCenter Portal

Use with WLST: Online

10.16.6.1 Description

Creates a federation trusted entity on an Oracle Secure Enterprise Search (SES) instance for a given entity name and password.

10.16.6.2 Syntax

```
createFederationTrustedEntity(appName, sesUrl, sesPassword, entityName,
entityPassword, desc,  [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application for which to perform this operation.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).

Argument	Definition
entityName	Entity name.
entityPassword	Entity password.
desc	Short description of the entity. Alternatively, specify an empty string ''.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.16.6.3 Example

The following example creates a federation trusted entity named myentity on the Oracle SES instance http://myseshost.com:7777:

```
wls:/weblogic/serverConfig> createFederationTrustedEntity(appName='webcenter',
sesUrl='http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', entityName='myentity', entityPassword='password',
desc='This is a sample entity')
```

10.17 Search - Oracle SES Search Crawlers

Use the commands listed in Table 10–24 to manage Oracle Secure Enterprise Search (SES) crawlers for WebCenter Portal applications.

There is no need to restart your WebCenter Portal application after running crawler WLST commands.

Table 10–24 Search - Oracle SES Crawler WLST Commands

Use this command...	To...	Use with WLST...
createSpacesCrawler	Create a crawler for Spaces objects on an Oracle SES instance.	Online
createDocumentsCrawler	Create a documents crawler for a WebCenter Portal application, on an Oracle SES instance.	Online
createDiscussionsCrawler	Create a discussions crawlers and an announcement crawler for a WebCenter Portal application, on an Oracle SES instance.	Online
listSpacesCrawler	Return the Spaces crawler configured for a Spaces application, on an Oracle SES instance.	Online
listDocumentsCrawler	Return the documents crawler configured for a WebCenter Portal application, on an Oracle SES instance.	Online
listDiscussionsCrawler	Return the discussion and announcement crawlers configured for a WebCenter Portal application, on an Oracle SES instance.	Online
startSpacesCrawler	Start the Spaces crawler configured for a Spaces application, on an Oracle SES instance.	Online

Table 10–24 (Cont.) Search - Oracle SES Crawler WLST Commands

Use this command...	To...	Use with WLST...
startDocumentsCrawler	Start the documents crawler configured for a WebCenter Portal application, on an Oracle SES instance.	Online
startDiscussionsCrawler	Start the discussion and announcement crawlers configured for a WebCenter Portal application, on an Oracle SES instance.	Online
stopSpacesCrawler	Stop the Spaces crawler configured for a Spaces application, on an Oracle SES instance.	Online
stopDocumentsCrawler	Stop the documents crawler configured for a WebCenter Portal application, on an Oracle SES instance.	Online
stopDiscussionsCrawler	Stop discussion and announcement crawlers configured for a WebCenter Portal application, on an Oracle SES instance.	Online
deleteSpacesCrawler	Delete the Spaces crawler configured for a Spaces application, on an Oracle SES instance.	Online
deleteDocumentsCrawler	Delete the documents crawler configured for a WebCenter Portal application, on an Oracle SES instance.	Online
deleteDiscussionsCrawler	Delete discussion and announcement crawlers configured for a WebCenter Portal application, on an Oracle SES instance.	Online

10.17.1 createSpacesCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.1.1 Description

Creates a crawler for Spaces objects on an Oracle SES instance. The command creates a WebCenter Portal datasource and specifies a schedule for crawling Spaces objects (such as spaces, lists, pages, and people).

10.17.1.2 Syntax

```
createSpacesCrawler(appName,  host, port, sesUrl, sesPassword, crawlUser,
crawlPassword, scratchDir, authUserIdFormat, crawlingMode, recrawlPolicy,
freqType, startHour, hoursBetweenLaunches, startDayOfWeek, startDayOfMonth,
daysBetweenLaunches, weeksBetweenLaunches, monthsBetweenLaunches, [server,
applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
host	Host name of the machine where the Spaces application is running.
port	Port number used to access the Spaces application.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`

Argument	Definition
sesPassword	Password for the Oracle SES administrative user (eqsys).
crawlUser	Crawl administration user in the Spaces application. This user must exist in the Spaces application and in your back-end identity management server with appropriate permissions and roles. For example: mycrawladmin)
crawlPassword	Password for the Spaces user that is specified in the crawlUser argument.
scratchDir	Local directory where Oracle SES can write temporary status logs. The directory must be on the system where Oracle SES is installed.
authUserIdFormat	Format of the user ID in the active identity plug-in. For example, username, email, nickname, user_name.
crawlingMode	Mode for crawling URLs in the source. The default mode is ACCEPT_ALL. Valid values are: ACCEPT_ALL, INDEX_ONLY, EXAMINE_URL: **ACCEPT_ALL**—Automatically Accept All URLs for Indexing: Crawls and indexes all URLs in the source. It also extracts and indexes any links found in those URLs. Previously crawled URLs are only reindexed if they have changed. **EXAMINE_URL**—Examine URLs Before Indexing: Crawls but does not index any URLs in the source. It also crawls any links found in those URLs. **INDEX_ONLY**—Index Only: Crawls and indexes all URLs in the source. It does not extract any links found in those URLs. Select this option for a source previously crawled using **EXAMINE_URL**.
recrawlPolicy	Specifies whether to crawl all documents or only documents that have changed. Valid values are PROCESS_ALL and PROCESS_CHANGED: **PROCESS_ALL**—All documents are crawled. Use this option to force a full crawl. **PROCESS_CHANGED**—Only crawl documents that have changed since the previous crawl. This setting can significantly speed up the crawling process.
freqType	Frequency of scheduled crawls. Valid values are: MANUAL, MONTHLY, WEEKLY, DAILY, HOURLY. To schedule crawls MONTHLY, WEEKLY, DAILY, or HOURLY, specify additional arguments as follows: **MONTHLY**: startHour, startDayOfTheMonth, monthsBetweenLaunches **WEEKLY**: startHour, startDayOfTheWeek, weeksBetweenLaunches **DAILY**: startHour, daysBetweenLaunches **HOURLY:** hoursBetweenLaunches If regular crawls are not required, choose MANUAL and then use the startSpacesCrawler command to initiate a crawl manually.
startHour	Time to start the crawl. Any number between 1 and 24. For example, enter 2 for 2:00am, 14 for 2:00pm, and so on.
hoursBetweenLaunches	Number of hours between crawls. Only valid when freqType='HOURLY'.
startDayOfWeek	Day on which to start a weekly crawl. For example, MONDAY, TUESDAY, and so on. Only valid when freqType='WEEKLY'.

Argument	Definition
startDayOfMonth	Day of the month on which to start a monthly crawl. For example, enter 1 for 1st day of the month, 2 for 2nd day of the month, and so on. Only valid when `freqType='MONTHLY'`.
daysBetweenLaunches	Number of days between crawls. Only valid when `freqType='DAILY'`.
weeksBetweenLaunches	Number of weeks between crawls. Only valid when `freqType='WEEKLY'`.
monthsBetweenLaunches	Number of months between crawls. Only valid when `freqType='MONTHLY'`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.1.3 Example

The following example creates a Spaces crawler on the SES instance `http://myseshost.com:7777` for a Spaces application (`webcenter`) located at `http://myhost.com:8888/webcenter/spaces`:

```
createSpacesCrawler(appName='webcenter', host='myhost.com', port='8888',
sesUrl='http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='sespassword', crawlUser='mycrawladmin', crawlPassword='password',
scratchDir'/tmp', authUserIdFormat='username', crawlingMode='ACCEPT_ALL',
recrawlPolicy= 'PROCESS_ALL', freqType='MANUAL', startHour=1,
hoursBetweenLaunches=1, startDayOfWeek='MONDAY', startDayOfMonth=1,
daysBetweenLaunches =1, weeksBetweenLaunches=1, monthsBetweenLaunches=1)
```

10.17.2 createDocumentsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.2.1 Description

Creates a documents crawler for a WebCenter Portal application, on an Oracle SES instance.

The command creates an Oracle WebCenter Content repository datasource and specifies a schedule for crawling documents in the Oracle WebCenter Content repository.

10.17.2.2 Syntax

```
createDocumentsCrawler(appName,  host, port, sesUrl, sesPassword, configUrl,
user, password, scratchDir, httpEndpoint, displayUrl, realm, authUserIdFormat,
pipelineName, crawlingMode, recrawlPolicy, freqType, startHour,
hoursBetweenLaunches, startDayOfWeek, startDayOfMonth,
daysBetweenLaunches, weeksBetweenLaunches, monthsBetweenLaunches,
[server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
configUrl	URL of the XML configuration file providing details of the source, such as the data feed type, location, security attributes, and so on. Use the URL format: `http://<host>:<port>/cs /idcplg? IdcService=SES_CRAWLER_DOWNLOAD_CONFIG &source=<sourcename>`
user	Administrative user for Oracle WebCenter Content's Content Server. For example, `sysadmin`. If the authentication type is Oracle SSO, then enter a user ID (and password) of a user in the identity management server fronted by Oracle SSO. This user must be granted the same permissions as `sysadmin`. If it is not possible to grant those permissions, then delete the "remote" user corresponding to this user in Content Server, and create a "local" version of the user (same name) in Content Server.
password	Password for the administrative user specified.
scratchDir	Local directory where Oracle SES can write temporary status logs. The directory must be on the system where Oracle SES is installed.
`httpEndpoint`	HTTP endpoint for Content Server authorization. For example: `http://<host>:<port>/idc/idcplg`
`displayUrl`	HTTP host information string to prefix the relative access URL to form the complete display URL. For example: `http://<host>:<port>/idc`
realm	Realm of the application serving the control and data feed. This parameter is relevant when the feeds are accessed over HTTP and is mandatory when the authentication type is `BASIC`. For example, `jazn.com`
authUserIdFormat	Format of the user ID (in active identity plug-in) that is used by Content Server authorization API. For example, username, email, nickname, user_name.
pipelineName	Document service pipeline created for this source in Oracle SES.
crawlingMode	Mode for crawling URLs in the source. The default mode is `ACCEPT_ ALL`. Valid values are: `ACCEPT_ALL`, `INDEX_ONLY`, `EXAMINE_URL`: **ACCEPT_ALL**—Automatically Accept All URLs for Indexing: Crawls and indexes all URLs in the source. It also extracts and indexes any links found in those URLs. Previously crawled URLs are only reindexed if they have changed. **EXAMINE_URL**—Examine URLs Before Indexing: Crawls but does not index any URLs in the source. It also crawls any links found in those URLs. **INDEX_ONLY**—Index Only: Crawls and indexes all URLs in the source. It does not extract any links found in those URLs. Select this option for a source previously crawled using **EXAMINE_URL**

Argument	Definition
recrawlPolicy	Specifies whether to crawl all documents or only documents that have changed. Valid values are PROCESS_ALL and PROCESS_CHANGED: **PROCESS_ALL**—All documents are crawled. Use this option to force a full crawl. **PROCESS_CHANGED**—Only crawl documents that have changed since the previous crawl. This setting can significantly speed up the crawling process.
freqType	Frequency of scheduled crawls. Valid values are: MANUAL, MONTHLY, WEEKLY, DAILY, HOURLY. To schedule crawls MONTHLY, WEEKLY, DAILY, or HOURLY, specify additional arguments as follows: **MONTHLY:** startHour, startDayOfTheMonth, monthsBetweenLaunches **WEEKLY**: startHour, startDayOfTheWeek, weeksBetweenLaunches **DAILY**: startHour, daysBetweenLaunches **HOURLY:** hoursBetweenLaunches If regular crawls are not required, choose MANUAL and then use the startDocumentsCrawler command to initiate a crawl manually.
startHour	Time to start the crawl. Any number between 1 and 24. For example, enter 2 for 2:00am, 14 for 2:00pm, and so on.
hoursBetweenLaunches	Number of hours between crawls. Only valid when freqType='HOURLY'.
startDayOfWeek	Day on which to start a weekly crawl. For example, MONDAY, TUESDAY, and so on. Only valid when freqType='WEEKLY'.
startDayOfMonth	Day of the month on which to start a monthly crawl. For example, enter 1 for 1st day of the month, 2 for 2nd day of the month, and so on. Only valid when freqType='MONTHLY'.
daysBetweenLaunches	Number of days between crawls. Only valid when freqType='DAILY'.
weeksBetweenLaunches	Number of weeks between crawls. Only valid when freqType='WEEKLY'.
monthsBetweenLaunches	Number of months between crawls. Only valid when freqType='MONTHLY'.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.17.2.3 Example

The following example creates a documents crawler on the Oracle SES instance http://myseshost.com:7777 for a Spaces application (webcenter) located at http://myhost.com:8888/webcenter/spaces:

```
createDocumentsCrawler(appName='webcenter', host='myhost.com', port='8888',
```

```
sesUrl='http://myseshost.com:7777/search/api/admin/AdminService',
sesPaswword='password',
configUrl='http://myucmhost.com:9044/cs/idcplg?IdcService=SES_CRAWLER_DOWNLOAD_
CONFIG&source=mysource',
user='adminuser', password='password', scratchDir='/scratch',
httpEndpoint='http://myucmhost.com:9044/cs/idcplg',
displayUrl='http://myucmhost:9044/cs', realm='jazn.com',
authUserIdFormat='username',
pipelineName='My UCM Pipeline', crawlingMode='ACCEPT_ALL',
recrawlPolicy='PROCESS_ALL', freqType='MANUAL', startHour=1,
hoursBetweenLaunches=1, startDayOfWeek='MONDAY', startDayOfMonth=1,
daysBetweenLaunches=1, weeksBetweenLaunches=1, monthsBetweenLaunches=1)
```

10.17.3 createDiscussionsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.3.1 Description

Creates a discussion forum crawler and an announcements crawler for a WebCenter Portal application, on an Oracle Secure Enterprise Search (SES) instance.

The command creates two Oracle SES database sources (one for discussion forums and one for announcements) and specifies a crawl schedule. The discussion forums source is named `<appname_host_port>_forums` with a view of `FORUMCRAWLER_VW`, and the announcements source is named `<appname_host_port>_announcements` with a view of `ANNOUNCEMENTS_VW`.

10.17.3.2 Syntax

```
createDiscussionsCrawler(appName,  host, port, sesUrl, sesPassword,
dbConnString, user, password, authUserIdFormat, crawlingMode,
recrawlPolicy, freqType, startHour, hoursBetweenLaunches, startDayOfWeek,
startDayOfMonth, daysBetweenLaunches, weeksBetweenLaunches,
monthsBetweenLaunches, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
dbConnString	Connection URL for the database on which WebCenter Portal's discussions server is installed. Use the format: **Oracle:** `jdbc:oracle:thin:@<host>:<port>/<oracle-sid>` **IBM DB2:** `jdbc:db2://<host>:<port>/<database_name>` **Microsoft SQL Server**: `jdbc:sqlserver://<host_or_IP_address>:<port>/<database_name>`

Argument	Definition
user	Administrative user for the database on which WebCenter Portal's discussions server is installed. **Oracle:** The user `MyPrefix_DISCUSSIONS_CRAWLER` is created during WebCenter Portal's discussions server installation. **IBM DB2**: The user `MyPrefix_DC` is created during WebCenter Portal's discussions server installation (where `MyPrefix` is five characters) **Microsoft SQL Server**: The user `MyPrefix_DISCUSSIONS_CRAWLER` is created during WebCenter Portal's discussions server installation.
password	Password for the administrative discussions server user specified.
authUserIdFormat	Format of the user ID (in active identity plug-in), that is used by the discussions server authorization API. For example, username, email, nickname, user_name.
crawlingMode	Mode for crawling URLs in the source. The default mode is `ACCEPT_ALL`. Valid values are: `ACCEPT_ALL`, `INDEX_ONLY`, `EXAMINE_URL`: **ACCEPT_ALL**—Automatically Accept All URLs for Indexing: Crawls and indexes all URLs in the source. It also extracts and indexes any links found in those URLs. Previously crawled URLs are only reindexed if they have changed. **EXAMINE_URL**—Examine URLs Before Indexing: Crawls but does not index any URLs in the source. It also crawls any links found in those URLs. **INDEX_ONLY**—Index Only: Crawls and indexes all URLs in the source. It does not extract any links found in those URLs. Select this option for a source previously crawled using **EXAMINE_URL**.
recrawlPolicy	Specifies whether to crawl all documents or only documents that have changed. Valid values are `PROCESS_ALL` and `PROCESS_CHANGED`: **PROCESS_ALL**—All documents are crawled. Use this option to force a full crawl. **PROCESS_CHANGED**—Only crawl documents that have changed since the previous crawl. This setting can significantly speed up the crawling process.
freqType	Frequency of scheduled crawls. Valid values are: `MANUAL`, `MONTHLY`, `WEEKLY`, `DAILY`, `HOURLY`. To schedule crawls MONTHLY, WEEKLY, DAILY, or HOURLY, specify additional arguments as follows: **MONTHLY:** startHour, startDayOfTheMonth, monthsBetweenLaunches **WEEKLY**: startHour, startDayOfTheWeek, weeksBetweenLaunches **DAILY**: startHour, daysBetweenLaunches **HOURLY:** hoursBetweenLaunches If regular crawls are not required, choose `MANUAL` and then use the startDiscussionsCrawler command to initiate a crawl manually.
startHour	Time to start the crawl. Any number between 1 and 24. For example, enter 2 for 2:00am, 14 for 2:00pm, and so on.
`hoursBetweenLaunches`	Number of hours between crawls. Only valid when `freqType='HOURLY'`.
startDayOfWeek	Day on which to start a weekly crawl. For example, MONDAY, TUESDAY, and so on. Only valid when `freqType='WEEKLY'`.

Argument	Definition
startDayOfMonth	Day of the month on which to start a monthly crawl. For example, enter 1 for 1st day of the month, 2 for 2nd day of the month, and so on. Only valid when `freqType='MONTHLY'`.
daysBetweenLaunches	Number of days between crawls. Only valid when `freqType='DAILY'`.
weeksBetweenLaunches	Number of weeks between crawls. Only valid when `freqType='WEEKLY'`.
monthsBetweenLaunches	Number of months between crawls. Only valid when `freqType='MONTHLY'`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.3.3 Example

The following example creates a discussion forum crawler and an announcements crawler on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application (`webcenter`) located at `http://myhost.com:8888/webcenter/spaces`:

```
createDiscussionsCrawler(appName='webcenter', host='myhost.com', port='8888',
sesUrl='http://myseshost.com:7777/search/api/admin/AdminService',
sesPaswword='password',
dbConnString='jdbc:oracle:thin:@myjivedbhost.com:1521/mysid',
user='app_discussions_crawler', password='password',
 authUserIdFormat='nickname', crawlingMode='ACCEPT_ALL',
recrawlPolicy='PROCESS_ALL', freqType='MANUAL', startHour=1,
 hoursBetweenLaunches=1,startDayOfWeek='MONDAY',
startDayOfMonth=1, daysBetweenLaunches=1,
weeksBetweenLaunches=1, monthsBetweenLaunches=1)
```

10.17.4 listSpacesCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.4.1 Description

Returns the Spaces crawler configured for a Spaces application, on an Oracle SES instance.

10.17.4.2 Syntax

```
listSpacesCrawler(appName, sesUrl, sesPassword, host, port, [verbose],
[server], [applicationVersion]
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the Spaces application is running.
port	Port number used to access the Spaces application.
verbose	Optional. Valid options are `1` (true) and `0` (false). When set to `1`, `listSpacesCrawlers` returns the Spaces crawler configured for a Spaces application in Oracle SES, along with details. When set to `0`, only source names are listed. This argument defaults to `0`.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Portal application is deployed.

10.17.4.3 Example

The following example returns the Spaces crawler configured in the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
listSpacesCrawler(appName='webcenter',
sesUrl='http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword'password', host='myhost.com', port='8888')

Already in Domain Runtime Tree
-----------------
Spaces Crawlers
-----------------
webcenter_myhost.com_8888_spaces
```

10.17.5 listDocumentsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.5.1 Description

Returns the document crawler configured for a WebCenter Portal application, on an Oracle SES instance.

10.17.5.2 Syntax

```
listDocumentsCrawler(appName, sesUrl, sesPassword, host, port, [verbose],
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

Argument	Definition
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.
verbose	Optional. Valid options are `1` (true) and `0` (false). When set to `1`, `listDocumentsCrawlers` returns the documents crawler that is configured for a WebCenter Portal application in Oracle SES, along with details. When set to `0`, only source names are listed. This argument defaults to `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.5.3 Example

The following example returns the documents crawler configured in the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
listDocumentsCrawler(appName='webcenter',
sesUrl='http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword'password', host='myhost.com', port='8888')

Already in Domain Runtime Tree
-----------------
Documents Crawlers
-----------------
webcenter_myhost.com_8888_documents
```

10.17.6 listDiscussionsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.6.1 Description

Returns the discussion and announcement crawlers configured for a WebCenter Portal application, on an Oracle SES instance.

10.17.6.2 Syntax

```
listDiscussionsCrawler(appName, sesUrl, sesPassword, host, port,
[verbose],  [server], [applicationVersion]
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

Argument	Definition
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.
verbose	Optional. Valid options are `1` (true) and `0` (false). When set to `1`, `listDocumentsCrawlers` returns discussion and announcement crawlers that are configured for a WebCenter Portal application in Oracle SES, along with details. When set to `0`, only source names are listed. This argument defaults to `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.6.3 Example

The following example returns discussion and announcement crawlers configured in the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
listDiscussionsCrawler(appName='webcenter',
sesUrl='http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')

Already in Domain Runtime Tree
-----------------
Discussions Crawler
-----------------
webcenter_myhost.com_8888_forums
webcenter_myhost.com_8888_announcements
```

10.17.7 startSpacesCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.7.1 Description

Starts the Spaces crawler configured for a Spaces application, on an Oracle SES instance.

10.17.7.2 Syntax

```
startSpacesCrawler(appName, sesUrl, sesPassword, host, port, [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the Spaces application is running.
port	Port number used to access the Spaces application.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.17.7.3 Example

The following example starts the Spaces crawler configured on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
startSpacesCrawler(appName='webcenter',
sesUrl'http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')
```

10.17.8 startDocumentsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.8.1 Description

Starts the documents crawler configured for a WebCenter Portal application, on an Oracle SES instance.

10.17.8.2 Syntax

```
startDocumentsCrawler(appName, sesUrl, sesPassword, host, port, [server],
[applicationVersion]
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.8.3 Example

The following example starts the document crawler configured on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
startDocumentsCrawler(appName='webcenter',
sesUrl'http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')
```

10.17.9 startDiscussionsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.9.1 Description

Starts the discussion and announcement crawlers configured for a WebCenter Portal application, on an Oracle SES instance.

10.17.9.2 Syntax

```
startDiscussionsCrawler(appName, sesUrl, sesPassword, host, port,  [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.9.3 Example

The following example starts the discussion and announcement crawlers configured on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
startDiscussionsCrawler(appName='webcenter',
sesUrl'http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')
```

10.17.10 stopSpacesCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.10.1 Description

Stops the Spaces crawler configured for a Spaces application, on an Oracle SES instance.

10.17.10.2 Syntax

```
stopSpacesCrawler(appName, sesUrl, sesPassword, host, port, [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the Spaces application is running.
port	Port number used to access the Spaces application.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.17.10.3 Example

The following example stops the Spaces crawler configured on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
stopSpacesCrawler(appName='webcenter',
sesUrl'http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')
```

10.17.11 stopDocumentsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.11.1 Description

Stops the documents crawler configured for a WebCenter Portal application, on an Oracle SES instance.

10.17.11.2 Syntax

```
stopDocumentsCrawler(appName, sesUrl, sesPassword, host, port, [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.11.3 Example

The following example stops the document crawler configured on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
stopDocumentsCrawler(appName='webcenter',
sesUrl'http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')
```

10.17.12 stopDiscussionsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.12.1 Description

Stops the discussion and announcement crawlers configured for a WebCenter Portal application, on an Oracle SES instance.

10.17.12.2 Syntax

```
stopDiscussionsCrawler(appName, sesUrl, sesPassword, host, port,
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

Argument	Definition
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.12.3 Example

The following example stops the discussion and announcement crawlers configured on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
stopDiscussionsCrawler(appName='webcenter',
sesUrl'http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')
```

10.17.13 deleteSpacesCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.13.1 Description

Deletes the Spaces crawler configured for a Spaces application, on an Oracle SES instance.

10.17.13.2 Syntax

```
deleteSpacesCrawler(appName, sesUrl, sesPassword, host, port,[server],
[applicationVersion])
```

Argument	Definition
appName	Name of the application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the Spaces application is running.
port	Port number used to access the Spaces application.

Argument	Definition
server	Optional. Name of the managed server where the application is deployed. For example, `WC_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 application is deployed.

10.17.13.3 Example

The following example deletes the Spaces crawler configured on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
deleteSpacesCrawler(appName='webcenter',
sesUrl'http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')
```

10.17.14 deleteDocumentsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.14.1 Description

Deletes the documents crawler configured for a WebCenter Portal application, on an Oracle SES instance.

10.17.14.2 Syntax

```
deleteDocumentsCrawler(appName, sesUrl, sesPassword, host, port, [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.14.3 Example

The following example deletes the document crawler configured on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
deleteDocumentsCrawler(appName='webcenter',
sesUrl'http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')
```

10.17.15 deleteDiscussionsCrawler

Module: Oracle WebCenter Portal

Use with WLST: Online

10.17.15.1 Description

Deletes the discussion and announcement crawlers configured for a WebCenter Portal application, on an Oracle SES instance.

10.17.15.2 Syntax

```
deleteDiscussionsCrawler(appName, sesUrl, sesPassword, host, port,  [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
sesUrl	Web Service URL for the Oracle SES Administration API. Use the format: `http://<host>:<port>/search/api/admin/AdminService`
sesPassword	Password for the Oracle SES administrative user (`eqsys`).
host	Host name of the machine where the WebCenter Portal application is running.
port	Port number used to access the WebCenter Portal application.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.17.15.3 Example

The following example deletes the discussion and announcement crawlers configured on the Oracle SES instance `http://myseshost.com:7777` for a Spaces application named `webcenter` located at `http://myhost.com:8888/webcenter/spaces`:

```
deleteDiscussionsCrawler(appName='webcenter',
sesUrl'http://myseshost.com:7777/search/api/admin/AdminService',
sesPassword='password', host='myhost.com', port='8888')
```

10.18 Search - WebCenter Portal Search

Use the commands listed in Table 10–25 to manage search settings and crawl options for the Spaces application and other WebCenter Portal applications.

Configuration changes made using these WebCenter Portal WLST commands are effective immediately; no restart is required.

Table 10–25 WebCenter Portal Search WLST Commands

Use This Command...	To...	Use with WLST...
setSearchConfig	Modify search settings for a WebCenter Portal application.	Online
listSearchConfig	List search properties for a WebCenter Portal application.	Online
setSpacesCrawlProperties	Specify crawl properties for a WebCenter Portal application.	Online
getSpacesCrawlProperties	Return the current crawl settings for a WebCenter Portal application.	Online

10.18.1 setSearchConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.18.1.1 Description

Modifies search settings for a WebCenter Portal application. If a parameter is not specified it is not modified.

10.18.1.2 Syntax

```
setSearchConfig(appName,[numSavedSearches],[numResultsRegion],[numResultsMain],
[executionTimeout],[prepareTimeout],[showAllExecutionTimeout],
[server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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.
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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.18.1.3 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.

```
wls:/weblogic/serverConfig> setSearchConfig(appName='webcenter',
numSavedSearches=8);
```

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);
```

10.18.2 listSearchConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.18.2.1 Description

Lists search settings for a WebCenter Portal application.

10.18.2.2 Syntax

```
listSearchConfig(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application for which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.18.2.3 Example

The following example displays search configuration information for a WebCenter Portal application named `webcenter`.

```
wls:/weblogic/serverConfig> listSearchConfig(appName='webcenter')
```

10.18.3 setSpacesCrawlProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.18.3.1 Description

Specifies crawl properties for WebCenter Portal applications.

WebCenter Portal applications can be crawled by Oracle SES to provide a faster, more unified search experience across WebCenter Portal objects, specifically: spaces, lists, pages, people (profiles), wikis, blogs, documents, discussions, and announcements. Three distinct crawlers make this possible:

- Spaces Crawler (for spaces, lists, pages, and people)
- Documents Crawler (for documents, wikis, blogs)
- Discussions Crawler (for discussions and announcements).

Use this command to enable or disable Oracle SES crawlers in WebCenter Portal applications:

- **Spaces application**—To use Oracle SES crawlers in the Spaces application, you *must* enable all three crawlers.
- **Framework applications**—To use Oracle SES crawlers in WebCenter Portal applications, you *must* enable both the documents and discussions crawlers. The Spaces crawler is not applicable.

(Spaces application only) You can also use this command to specify an interval between full crawls for the Spaces crawler. During a full crawl, all of the Spaces crawler content is re-read. Out-of-the-box, full crawls for the Spaces crawler occur every seven days but you can specify a different frequency to suit your installation.

Note that incremental crawls, for all three crawlers, are initiated by a scheduler running from Oracle SES. During these incremental crawls, only content added or updated since the previous crawl is processed.

10.18.3.2 Syntax

```
setSpacesCrawlProperties(appName, [fullCrawlIntervalInHours, spacesCrawlEnabled,
documentCrawlEnabled, discussionsCrawlEnabled, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application.
fullCrawlIntervalInHours	Optional. (Spaces application only) Number of hours between full crawls. The default is 168 hours or 7 days.
spacesCrawlEnabled	Optional. Specifies whether the Spaces Crawler is enabled in Oracle SES. Valid values are `1` (true) and `0` (false). This argument defaults to `0`. When set to `0`, WebCenter Portal's internal search adapters return search results.
documentCrawlEnabled	Optional. Specifies whether the Documents Crawler is enabled in Oracle SES. Valid values are `1` (true) and `0` (false). This argument defaults to `0`. When set to `0`, WebCenter Portal's internal search adapters return search results.

Argument	Definition
discussionsCrawlEnabled	Optional. Specifies whether the Discussions Crawler is enabled in Oracle SES. Valid values are `1` (true) and `0` (false). This argument defaults to `0`. When set to `0`, WebCenter Portal's internal search adapters return search results.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.18.3.3 Example

The following example enables Oracle SES crawlers in the Spaces application and specifies that the Spaces application runs a full crawl through the Spaces Crawler every 8 days:

```
wls:/weblogic/serverConfig>setSpacesCrawlProperties(appName='webcenter',
fullCrawlIntervalInHours=192, spacesCrawlEnabled=1, documentCrawlEnabled=1,
discussionsCrawlEnabled=1)
```

10.18.4 getSpacesCrawlProperties

Module: Oracle WebCenter Portal

Use with WLST: Online

10.18.4.1 Description

Returns the current crawl settings for a WebCenter Portal application, such as the number of hours between full crawls (Spaces crawler), and whether Oracle SES crawlers are enabled.

10.18.4.2 Syntax

```
getSpacesCrawlProperties(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.18.4.3 Example

The following example returns the current crawl settings for the Spaces application.

```
wls:/weblogic/serverConfig>getSpacesCrawlProperties(appName='webcenter')

Spaces Crawl Properties:
```

```
-----------------
fullCrawlIntervalInHours: 124
spacesCrawlEnabled:       1
documentCrawlEnabled:     1
discussionsCrawlEnabled:  1
```

10.19 Worklists

Use the commands listed in Table 10–26 to manage BPEL server connections for WebCenter Portal applications.

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

Table 10–26 Worklist Commands

Use this command...	To...	Use with WLST...
createBPELConnection	Create a connection to a BPEL server for a WebCenter Portal application.	Online
setBPELConnection	Edit an existing BPEL server connection.	Online
listBPELConnections	List all of the BPEL server connections that are configured for a WebCenter Portal 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

10.19.1 createBPELConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.19.1.1 Description

Creates a connection to a BPEL server for a named WebCenter Portal application. BPEL server connections can be used by the application's Worklist service and Spaces workflows.

To configure the Worklist service to actively use a new BPEL server connection, use the `addWorklistConnection` command. See Section 10.19.4, "addWorklistConnection".

To specify the BPEL server connection that Spaces uses for its internal workflows, use the `setSpacesWorkflowConnectionName` command. See Section 10.20.2, "setSpacesWorkflowConnectionName".

10.19.1.2 Syntax

```
createBPELConnection(appName, name, url, [policy, recipientKeyAlias, linkUrl,
server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Connection name. The name must be unique (across all connection types) within the WebCenter Portal 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 Portal application.
policy	Optional. SAML token policy this connection uses for authentication. Enter any valid policy. Valid values include: ■ oracle/wss10_saml_token_client_policy—use to access the BPEL server with the default, non message protected policy. ■ oracle/wss10_saml_token_with_message_protection_client_policy—use to access the BPEL server with a message protected policy. If selected, you must configure keys stores both in your WebCenter Portal application and in the BPEL application. ■ GPA—use if your environment supports Global Policy Attachments (GPA). If you omit this argument, the connection defaults to oracle/wss10_saml_token_client_policy.
recipientKeyAlias	Optional. Recipient key alias to be used for message protected SAML policy authentication. Only required when the BPEL server connection is using a SAML token policy for authentication and the application's Worklist service is using multiple BPEL server connections. The default is null. See also "Configuring WS-Security for WebCenter Portal Applications and Components" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
linkUrl	Optional. URL used to link to the BPEL server. Only required if it is different to the url argument. For example, when SSO or HTTPS is configured. Use the format: *protocol://host:port* The default is null. For performance reasons, in an HTTPS or SSO environment, linkUrl specifies user access to BPEL worklist items, through HTTPS or SSO Web servers, whereas url specifies direct access to BPEL Web services, without redirection through HTTPS or SSO Web servers.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.19.1.3 Examples

The following example creates a connection named WebCenter Worklist with the default security policy:

```
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 a message protected security policy, and defines a specific link URL:

```
wls:/weblogic/serverConfig> createBPELConnection(appName='webcenter',
name='WebCenter Worklist',url='http://myhost.com:8001', policy='oracle/wss10_
saml_token_with_message_protection_client_policy', recipientKeyAlias='myalias',
linkUrl='http://mySSO.com:7777')
```

The following example creates a connection to be used in an environment that supports Global Policy Attachments (GPA):

```
wls:/weblogic/serverConfig> createBPELConnection(appName='webcenter',
name='WebCenter Worklist', url='http://myhost.com:8001' policy='GPA')
```

10.19.2 setBPELConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.19.2.1 Description

Edits an existing BPEL server connection.

To configure the Worklist service to actively use an existing BPEL server connection, use the `addWorklistConnection` command. See Section 10.19.4, "addWorklistConnection".

To specify the BPEL server connection used for Webcenter Spaces workflows, use the `setSpacesWorkflowConnectionName` command. See Section 10.20.2, "setSpacesWorkflowConnectionName".

10.19.2.2 Syntax

```
setBPELConnection(appName, name, [url, policy, recipientKeyAlias, linkUrl, server,
applicationVersion])
```

Argument	Definition
`appName`	Name of the WebCenter Portal application in which 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 Portal application.

Argument	Definition
policy	Optional. SAML token policy this connection uses for authentication. Enter any valid policy. Valid values include: ■ oracle/wss10_saml_token_client_policy—use to access the BPEL server with the default, non message protected policy. ■ oracle/wss10_saml_token_with_message_protection_client_policy—use to access the BPEL server with a message protected policy. If selected, you must configure keys stores both in your WebCenter Portal application and in the BPEL application. ■ GPA—use if your environment supports Global Policy Attachments (GPA). If you omit this argument, the connection defaults to oracle/wss10_saml_token_client_policy.
recipientKeyAlias	Optional. Recipient key alias to be used for message protected SAML policy authentication. Only required when the BPEL server connection is using a SAML token policy for authentication and the application's Worklist service is using multiple BPEL server connections. The default is null. See also "Configuring WS-Security for WebCenter Portal Applications and Components" in the *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
linkUrl	Optional. URL used to link to the BPEL server. Only required if it is different to the url argument. For example, when SSO or https is configured. Use the format: *protocol://host:port* For example, http://mySSO.host.com:7777 The default is null. For performance reasons, in an HTTPS or SSO environment, the **Link URL** specifies user access to BPEL worklist items, through HTTPS or SSO Web servers, whereas the **BPEL SOAP URL** specifies direct access to BPEL Web services, without redirection through HTTPS or SSO Web servers.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, WC_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 Portal application is deployed.

10.19.2.3 Examples

The following example updates the BPEL server URL, security policy, recipient key alias, and link url 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', recipientKeyAlias='myalias',
linkUrl='http://mySSO.com:7777')
```

The following example changes the security policy to use Global Policy Attachments (GPA):

```
wls:/weblogic/serverConfig> setBPELConnection(appName='webcenter',
name='WebCenter Worklist', policy='GPA')
```

10.19.3 listBPELConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.19.3.1 Description

Without any arguments, this command lists all the BPEL connections that are configured for a specific WebCenter Portal application. All BPEL connections are listed, even connections not currently used.

10.19.3.2 Syntax

```
listBPELConnections(appName, [verbose], [name], [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application for which to list BPEL server connections.
verbose	Optional. Displays BPEL server connection details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listBPELConnections` lists all of the BPEL server connections that are configured, along with their details. When set to `0`, `listBPELConnections` lists connection names only. This argument defaults to `0`. If you set this argument to `0`, 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.19.3.3 Examples

The following example lists the names of all the BPEL server connections that are configured for a WebCenter Portal 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 Portal application.

```
wls:/weblogic/serverConfig> listBPELConnections(appName='webcenter', verbose=1)
  ------------------
  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
  ------------------
```

10.19.4 addWorklistConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.19.4.1 Description

Enable an existing BPEL server connection for Worklist services. The Worklist service supports multiple connections so that WebCenter Portal users can monitor and manage assignments and notifications from a range of BPEL servers.

The name must specify an existing BPEL server connection.

10.19.4.2 Syntax

```
addWorklistConnection(appName, name,[verbose, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which 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. `1` turns verbose mode on; `0` turns verbose mode off. This argument defaults to `0`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.19.4.3 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=1)
  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=1)
  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
```

10.19.5 removeWorklistConnection

Module: Oracle WebCenter Portal

Use with WLST: Online

10.19.5.1 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.

10.19.5.2 Syntax

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

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
name	Name of an existing BPEL server connection.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.19.5.3 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
```

10.19.6 listWorklistConnections

Module: Oracle WebCenter Portal

Use with WLST: Online

10.19.6.1 Description

Without any arguments, this command lists all of the BPEL server connections that are configured for the Worklist service, in a named WebCenter Portal application.

10.19.6.2 Syntax

```
listWorklistConnections(appName, [verbose],[name], [server],
[applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application for which to perform this operation.
verbose	Optional. Displays BPEL server connection details in verbose mode. Valid options are `1` (true) and `0` (false). When set to `1`, `listWorklistConnections` lists all of the BPEL server connections that are configured for the Worklist service, along with their details. When set to `0`, `listWorklistConnections` lists connection names only. This argument defaults to `0`. If you set this argument to `0`, 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 Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.19.6.3 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=1)
  ------------------
  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=1, name='MyWorklist')
  ------------------
  The following connection is not in the ADF Worklist:MyWorklist
```

10.20 Spaces Application

Use the commands listed in Table 10–27 to manage workflow settings and metadata for the Spaces application.

Table 10–27 Spaces Application WLST Commands

Use This Command...	To...	Use with WLST...
getSpacesWorkflowConnectionName	Return the name of the BPEL server connection that the Spaces application is using for internal workflows.	Online
setSpacesWorkflowConnectionName	Specify the BPEL server connection used for Spaces workflows.	Online
refreshGroupSpaceCache	Migrate metadata for individual spaces (in MDS) and space security data to the 'Spaces Cache'.	Online
refreshSpaceTemplateCache	Migrate metadata for individual space templates (in MDS) and space template security data to the 'Space Templates Cache'.	Online

10.20.1 getSpacesWorkflowConnectionName

Module: Oracle WebCenter Portal

Use with WLST: Online

10.20.1.1 Description

Returns the name of the BPEL server connection that the Spaces application is currently using for internal workflows (spacemembership notifications, spacesubscription requests, and so on).

10.20.1.2 Syntax

```
getSpacesWorkflowConnectionName(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application—always `webcenter`.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.20.1.3 Example

The following example names the BPEL server connection that the Spaces application is currently using for internal workflow.

```
wls:/weblogic/serverConfig> getSpacesWorkflowConnectionName(appName='webcenter')
WorkflowConfigConnectionName: WebCenter-Worklist
```

10.20.2 setSpacesWorkflowConnectionName

Module: Oracle WebCenter Portal

Use with WLST: Online

10.20.2.1 Description

Specifies the BPEL server connection that the Spaces application uses for internal workflows. The Spaces application uses a BPEL server included with the Oracle SOA Suite to host internal workflows, such as spacemembership notifications, spacesubscription requests, and so on. The connection name specified here must be a valid BPEL server connection.

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

10.20.2.2 Syntax

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

Argument	Definition
appName	Name of the Spaces application—always `webcenter`.
name	Name of an existing BPEL connection.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.20.2.3 Example

The following example specifies that the Spaces application uses the BPEL server connection named `WebCenter-Worklist` for its internal workflows.

```
wls:/weblogic/serverConfig>setSpacesWorkflowConnectionName(appName='webcenter',
name='WebCenter-Worklist')
```

10.20.3 refreshGroupSpaceCache

Module: Oracle WebCenter Portal

Use with WLST: Online

10.20.3.1 Description

Migrates metadata for individual spaces (in MDS) and space security data (in a policy store) to the '*Spaces Cache*'.

WebCenter Spaces 11.1.1.2.0 (and later) uses tables (referred to as the Spaces Cache) to store space metadata and security-related data. When you migrate from WebCenter Spaces 11.1.1.1.0 to 11.1.1.2.0, you must run the `refreshGroupSpaceCache` command so that all your existing space data is migrated to the new 'Spaces Cache'.

10.20.3.2 Syntax

```
refreshGroupSpaceCache(appName, [spaceNames, syncMode, updateType, cleanCache])
```

Argument	Definition
appName	Name of the Spaces application—always `webcenter`.
spaceNames	Optional. Names of one or more spaces (group spaces) that you want to refresh. To refresh all the spaces in MDS, enter `spaceNames= ''` To refresh selective spaces, enter one or more space names separated with a comma, for example: spaceNames='`MyGroupSpace1,MyGroupSpace`'
updateType	Optional. Indicates the type of data to refresh. Valid values are: `security`, `metadata`, `all`. The default value is `security`. ■ `security` - Refreshes the cache with security data stored in the policy store. The security data that is stored includes member data for the space, whether or not a space is public, and whether or not a space is accessible to users assigned the `Authenticated-User` role (in earlier releases this role was named `Spaces-User`). ■ `metadata` - Refreshes the cache with space-related metadata stored in MDS. The data that is stored includes metadata information such as the display name, keywords, icon, logo, and so on. ■ `all` - Refreshes the cache with data stored in MDS and the policy store.
syncMode	Optional. Indicates whether to refresh the Spaces application in synchronous or asynchronous mode. Valid values are `1` and `0`. The default value is `1`. When set to `1`, the refresh process runs in synchronous mode. When set to `0`, the refresh is asynchronous, that is, a new thread is spawned for the refresh process. Synchronous mode is recommended.
cleanCache	Optional. Indicates whether to clear the Spaces Cache. Valid values are `1` and `0`. The default value is `0`. When set to `0`, the content of the Spaces application is not cleared during the refresh operation. Always set this value to `0` for migration. Use the `1` value only when importing an entire application, in which case the entire data available in the Spaces Cache is overwritten.

10.20.3.3 Example

The following examples update the cache to include all space-related metadata (in MDS) and security data (in the policy store) in synchronous mode:

```
wls:/weblogic/serverConfig>refreshGroupSpaceCache(appName='webcenter',
spaceNames='', syncMode=1, updateType='all', cleanCache=0)

wls:/weblogic/serverConfig>refreshGroupSpaceCache(appName='webcenter')
```

The following example updates the Spaces Cache to include space-related metadata (in MDS) and security data (in the policy store) for two spaces named `MyGroupSpace1` and `MyGroupSpace2`. The cache refreshes in synchronous mode.

```
wls:/weblogic/serverConfig>refreshGroupSpaceCache(appName='webcenter',
spaceNames='MyGroupSpace1,MyGroupSpace2')
```

10.20.4 refreshSpaceTemplateCache

Module: Oracle WebCenter Portal

Use with WLST: Online

10.20.4.1 Description

Migrates metadata for individual space templates (in MDS) and template security data (in a policy store) to the '*Space Templates Cache*'.

WebCenter Spaces 11.1.1.2.0 (and later) uses tables (referred to as the Space Templates Cache) to store space template metadata and security-related data. When you migrate from WebCenter Spaces 11.1.1.1.0 to 11.1.1.2.0, you must run the `refreshSpaceTemplateCache` command so that all your existing template data is migrated to the new 'Space Templates Cache'.

10.20.4.2 Syntax

```
refreshSpaceTemplateCache(appName, [spaceTemplateNames, syncMode, updateType,
cleanCache])
```

Argument	Definition
appName	Name of the Spaces application—always `webcenter`.
spaceNames	Optional. Names of one or more space templates that you want to refresh. To refresh all the space templates in MDS, enter `spaceTemplateNames= ''` To refresh selective space templates, enter one or more template names separated with a comma, for example: spaceNames='`MySpaceTemplate1,MySpaceTemplate`'
updateType	Optional. Indicates the type of data to refresh. Valid values are: `security`, `metadata`, `all`. The default value is `security`. ■ `security` - Refreshes the cache with security data stored in the policy store. The security data that is stored includes member data for the space template, whether or not a space template is public, and whether or not a space template is accessible to users assigned the `Authenticated-User` role (in earlier releases this role was named `Spaces-User`). ■ `metadata` - Refreshes the cache with space template-related metadata stored in MDS. The data that is stored includes metadata information such as the display name, keywords, icon, logo, and so on. ■ `all` - Refreshes the Space Template Cache with data stored in MDS and the policy store.
syncMode	Optional. Indicates whether to refresh the Spaces application in synchronous or asynchronous mode. Valid values are `1` and `0`. The default value is `1`. When set to `1`, the refresh process runs in synchronous mode. When set to `0`, the refresh is asynchronous, that is, a new thread is spawned for the refresh process. Synchronous mode is recommended.

Argument	Definition
cleanCache	Optional. Indicates whether to clear the Space Templates Cache. Valid values are `1` and `0`. The default value is `0`. When set to `0`, the content of the Spaces application is not cleared during the refresh operation. Always set this value to `0` for migration. Use the `1` value only when importing an entire application, in which case the entire data available in the Space Templates Cache is overwritten.

10.20.4.3 Example

The following examples update the cache to include all space template-related metadata (in MDS) and security data (in the policy store) in synchronous mode:

```
wls:/weblogic/serverConfig>refreshSpaceTemplateCache(appName='webcenter',
spaceTemplateNames='', syncMode=1, updateType='all', cleanCache=0)

wls:/weblogic/serverConfig>refreshSpaceTemplateCache(appName='webcenter')
```

The following example updates the Space Templates Cache to include space template related metadata (in MDS) and security data (in the policy store) for two space templates named `MySpaceTemplate1` and `MySpaceTemplate2`. The cache refreshes in synchronous mode.

```
wls:/weblogic/serverConfig>refreshSpaceTemplateCache(appName='webcenter',
spaceNames='MySpaceTemplate1,MySpaceTemplate2')
```

10.21 WebCenter Portal Identity Store

Use the commands listed in Table 10–28 to configure options for searching a WebCenter Portal application's identity store.

Table 10–28 WebCenter Portal Identity Store WLST Commands

Use this command...	To...	Use with WLST...
setWebCenterIdStoreSearchConfig	Modify configuration options for searching a WebCenter Portal applications's identity store.	Online
listWebCenterIdStoreSearchConfig	List current configuration options for searching a WebCenter Portal application's identity store.	Online
startSyncProfiles	Synchronize profile information in the LDAP store, with the WebCenter Portal database schema.	Online
stopSyncProfiles	Stop the profile synchronization process.	Online
isSyncProfilesRunning	Check whether profile synchronization is in progress.	Online
syncProfile	Synchronize profile information for a specific user.	Online
setProfileCacheNumberOfObjects	Set the number of profile objects to cache.	Online
setProfileSyncLDAPReadBatchSize	Set the profile synchronization LDAP batch read size.	Online
setProfileCacheTimeToLive	Set the time in minutes for profiles to live in profile cache.	Online

Table 10–28 (Cont.) WebCenter Portal Identity Store WLST Commands

Use this command...	To...	Use with WLST...
printProfileConfig	Print profile cache configuration values.	Online
renameUsersInWebCenterApplication	Rename users for a WebCenter Portal application.	Online
synchronizeUserInformation	Synchronize users in a WebCenter Portal application.	Online

10.21.1 setWebCenterIdStoreSearchConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.1.1 Description

Modifies configuration options for searching a WebCenter Portal application's identity store. Use these settings to optimize identity store searches (for users and roles) in a WebCenter Portal application.

Identity store search parameters are stored in `adf-config.xml`. If a search parameter is not specified, it is not modified.

10.21.1.2 Syntax

```
setWebCenterIdStoreSearchConfig(appName,[narrowSearchTimeout, broadSearchTimeout,
maxSearchFilters, maxFetchRecords, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
narrowSearchTimeout	Optional. Maximum time allowed (in ms) for small, simple searches, such as fetching a single user from the identity store. The out-of-the-box default is 30000ms.
broadSearchTimeout	Optional. Maximum time allowed (in ms) to return large result sets, such as returning users and roles that match a name pattern. The out-of-the-box default is 60000.
maxSearchFilters	Optional. Number of search filters allowed for the WebCenter Portal application's identity store. The maximum allowed, out-of-the-box, is 100. Some identity store searches are executed using search filters which are converted into LDAP search calls. If your associated LDAP server limits the search condition, you can set the `maxSearchFilters` property to match your LDAP server setting.
maxFetchRecords	Optional. Maximum number of records to be returned from each search query. The out-of-the-box default is `100`. The value of this setting will impact the performance of your LDAP server so take this into consideration when increasing the search result limit. Note that the LDAP server imposes its own search result limit, so the actual limit that is used will be the lesser of these two values.

Argument	Definition
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.21.1.3 Example

The following example increases both identity store search timeouts.

```
wls:/weblogic/serverConfig>setWebCenterIdStoreSearchConfig(appName='webcenter',
narrowSearchTimeout=60000, broadSearchTimeout=100000);
```

The following example limits the maximum number of records returned to 100.

```
wls:/weblogic/serverConfig>setWebCenterIdStoreSearchConfig(appName='webcenter',
maxFetchRecords=100);
```

10.21.2 listWebCenterIdStoreSearchConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.2.1 Description

Lists current configuration options for searching the WebCenter Portal application's identity store.

Identity store search parameters are stored in `adf-config.xml`.

10.21.2.2 Syntax

```
listWebCenterIdStoreSearchConfig(appName,[server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.21.2.3 Example

The following example displays identity store search configuration information for a WebCenter Portal application named `webcenter`.

```
wls:/weblogic/serverConfig>listWebCenterIdStoreSearchConfig(appName='webcenter');

-----------------
User role search configuration parameters
```

```
-----------------
Narrow search timeout    : 30000
Broad search timeout     : 60000
Maximum search filters   : 100
Maximum records to fetch : 200
```

10.21.3 startSyncProfiles

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.3.1 Description

Synchronizes profile information in the LDAP store, with the WebCenter Portal database schema.

10.21.3.2 Syntax

```
startSyncProfiles(appName)
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

10.21.3.3 Example

The following example synchronizes user profiles for an application named `webcenter`:

```
wls:/weblogic/serverConfig>startSyncProfiles(appName='webcenter')
```

10.21.4 stopSyncProfiles

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.4.1 Description

Stops the profile synchronization process, if currently in progress.

10.21.4.2 Syntax

```
stopSyncProfiles(appName)
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

10.21.4.3 Example

The following example stops the profile synchronization process for an application named `webcenter`:

```
wls:/weblogic/serverConfig>stopSyncProfiles(appName='webcenter')
```

10.21.5 isSyncProfilesRunning

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.5.1 Description

Checks whether profile synchronization is in progress.

10.21.5.2 Syntax

```
isSyncProfilesRunning(appName)
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

10.21.5.3 Example

The following example checks whether profile synchronization is in progress for an application named `webcenter`:

```
wls:/weblogic/serverConfig>isSyncProfilesRunning(appName='webcenter')
```

10.21.6 syncProfile

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.6.1 Description

Synchronizes profile information for a specific user in the LDAP store, with the WebCenter Portal schema.

10.21.6.2 Syntax

```
syncProfile(appName, userName)
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
userName	Name of the user whose profile information you want to synchronize.

10.21.6.3 Example

The following example synchronizes profile information for a user named `monty`:

```
wls:/weblogic/serverConfig>syncProfile(appName='webcenter', userName='monty')
```

10.21.7 setProfileCacheNumberOfObjects

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.7.1 Description

Sets the maximum number of profile objects to cache (in the profile cache).

10.21.7.2 Syntax

```
setProfileCacheNumberOfObjects(appName, noOfObjects)
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
noOfObjects	Number of profile objects to cache. The default value is `1000`.

10.21.7.3 Example

The following example increases the size of the cache to 2000 profiles:

```
wls:/weblogic/serverConfig>setProfileCacheNumberOfObjects(appName='webcenter',
noOfObjects=2000)
```

10.21.8 setProfileSyncLDAPReadBatchSize

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.8.1 Description

Sets the profile synchronization LDAP batch read size.

10.21.8.2 Syntax

```
setProfileSyncLDAPReadBatchSize(appName, batchSize)
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
batchSize	LDAP batch read size. The default value is `1000`.

10.21.8.3 Example

The following example increases the batch size to 2000 LDAP profiles:

```
wls:/weblogic/serverConfig>setProfileSyncLDAPReadBatchSize(appName='webcenter',
batchSize=2000)
```

10.21.9 setProfileCacheTimeToLive

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.9.1 Description

Sets the time in minutes for a profile to live in the profile cache.

10.21.9.2 Syntax

```
setProfileCacheTimeToLive(appName, timeToLive)
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
timeToLive	Time to live for profile objects (in minutes) in the profile cache. The default value is `60` minutes.

10.21.9.3 Example

The following example decreases the length of time profile objects are cached:

```
wls:/weblogic/serverConfig>setProfileCacheTimeToLive(appName='webcenter',
timeToLive=30)
```

10.21.10 printProfileConfig

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.10.1 Description

Prints profile cache configuration values.

10.21.10.2 Syntax

```
printProfileConfig(appName)
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.

10.21.10.3 Example

The following example displays the current profile cache configuration for an application named `webcenter`:

```
wls:/weblogic/serverConfig>printProfileConfig(appName='webcenter')
```

10.21.11 renameUsersInWebCenterApplication

Module: Oracle WebCenter Portal

Use with WLST: Online or Offline

10.21.11.1 Description

Renames incorrect user names in a WebCenter Portal application's policy store and WebCenter Portal application tables.

10.21.11.2 Syntax

```
renameUsersInWebCenterApplication(appName, names, component, [dbVendor,
dbHostNPort, dbName, dbUserName, verbose])
```

Note: You can run this command in offline mode for all the components except `PolicyStore`.

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
names	A comma separated list of users that need to be renamed. Use the format: `oldName=newName`
component	Specific the WebCenter Portal component in which users are to be renamed. Valid values are: `PolicyStore`, `Spaces`, `UCM`, and `JIVE`.
dbVendor	Optional. Database vendor. Valid values are: `Oracle`, `MSSQL`, and `IBMDB2`. Not required when `component=PolicyStore`.
dbHostNPort	Optional. Database host and port. Use the format: *`host:port`* Not required when `component=PolicyStore`.
dbName	Optional. Database name or *sid*. Not required when `component=PolicyStore`.
dbUserName	Optional. WebCenter Portal database schema name. Not required when `component=PolicyStore`.
verbose	Optional. Generates summary (`0`) or detailed output (`1`). Default value is `0`.

10.21.11.3 Example

This following example renames users in Spaces application tables:

```
wls:/weblogic/serverConfig>
renameUsersInWebCenterApplication(appName='webcenter',names='myOldname=myNewName,
myOldName1=myNewname1', component='Spaces', dbVendor='Oracle',
dbHostNPort='myDbHost.example.com:1521',dbName='myDb1',
dbUserName='webcenterUser1', verbose=1)
```

This following example renames users in the policy store for the Spaces application:

```
wls:/weblogic/serverConfig>
renameUsersInWebCenterApplication(appName='webcenter',names='myOldname=myNewName,
myOldName1=myNewname1', component='PolicyStore', verbose=1)
```

10.21.12 synchronizeUserInformation

Module: Oracle WebCenter Portal

Use with WLST: Online

10.21.12.1 Description

Synchronizes user details in a WebCenter Portal application's policy store and WebCenter Portal application tables.

10.21.12.2 Syntax

```
synchronizeUserInformation(appName, operationType, fileName, component, [dbVendor,
dbHostNPort, dbName, dbUserName, verbose])
```

Note: You can run this command in offline mode for all the components except `PolicyStore`.

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
operationType	Type of operation. Valid values are `RENAME_GUID`, `RENAME_USERNAME`, `DELETE_USER`: ■ `RENAME GUID` - Changes the GUID associated with one or more users ■ `RENAME_USERNAME` - Renames one or more users ■ `DELETE_USER` - Deletes one or more users
fileName	Fully qualified path to the file (including the file name) which contains the list of users to be modified or deleted.
component	WebCenter Portal component in which to rename or delete users. Valid values are: `PolicyStore`, `Spaces`, `UCM`, `DISCUSSION`, `ALL`.
dbVendor	Optional. Database vendor. Valid values are: `Oracle`, `MSSQL` and `IBMDB2`. Not required when `component=PolicyStore`.
dbHostNPort	Optional. Database host and port. Use the format: *`host:port`* Not required when `component=PolicyStore`.
dbName	Optional. Database name or *sid*. Not required when `component=PolicyStore`.
dbSchemaName	Optional. WebCenter Portal database schema name.
verbose	Optional. Generates summary (`0`) or detailed output (`1`). Default value is `0`.

10.21.12.3 Example

This following example renames WebCenter Portal users listed in `renamesusers.properties` as follows.

- `monty=monty1`
- `pat=pat1`

```
wls:/weblogic/serverConfig>
synchronizeUserInformation(appName='webcenter', operationType='RENAME_
USERNAME',fileName='/home/mydir/renameusers.properties', component='Spaces',
dbVendor='Oracle', dbHostNPort='myDbHost.example.com:1521',dbName='myDb1',
dbSchemaName='webcenterUser1', verbose=1)
```

This following example deletes WebCenter Portal user references for users listed in `delete.properties`.

- `monty=monty1`
- `pat=pat1`

```
wls:/weblogic/serverConfig>
synchronizeUserInformation(appName='webcenter',operationType='DELETE_
USERNAME',fileName='/home/mydir/delete.properties', component='Spaces',
dbVendor='Oracle', dbHostNPort='myDbHost.example.com:1521',dbName='myDb1',
dbSchemaName='webcenterUser1', verbose=1)
```

10.22 WebCenter Portal Import and Export

Use the commands listed in Table 10–29 to export and import the Spaces application and producer metadata associated with Framework applications.

Table 10–29 Import and Export WLST Commands

Use this command...	To...	Use with WLST...
exportWebCenterApplication	Export the Spaces application to an export archive.	Online
importWebCenterApplication	Import the Spaces application from an export archive.	Online
exportGroupSpaces	Export one or more spaces to an export archive.	Online
exportGroupSpaceTemplates	Export one or more space templates to an export archive.	Online
importGroupSpaces	Import one or more spaces or space templates from an export archive.	Online
setSpaceState	Take a space offline or brings a space online.	Online
exportWebCenterResource	Export a portal resource to an export archive (.EAR).	Online
importWebCenterResource	Import a portal resource from an export archive (.EAR).	Online
exportPortletClientMetadata	(Framework applications only.) Export portlet client metadata and producer customizations and personalizations to an export archive.	Online
importPortletClientMetadata	(Framework applications only.) Import portlet client metadata and producer customizations and personalizations from an export archive.	Online
importWebCenterTranslations	Import translations for the Spaces application.	Online
showProducerImportFailures	Display names of producers where metadata imports have failed and reasons for those failures	Online
retryAllFailedProducerImports	Attempt to import outstanding producer metadata	Online

10.22.1 exportWebCenterApplication

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.1.1 Description

(Spaces application only) Exports a Spaces application to an export archive (.EAR) using the filename provided.

10.22.1.2 Syntax

```
exportWebCenterApplication(appName, fileName, [exportCustomizations, exportData,
server, applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which 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 `1` (true) and `0` (false). When set to `1`, all application customizations are exported. When set to `0`, application customizations are not exported, that is, default task flows are exported without any customizations. This argument defaults to `1`.
exportData	Optional. Valid values are `1` (true) and `0` (false). When set to `1`, data stored in the Spaces database for activity streams, events, feedback, lists, links, message boards, people connections, profiles, and tags is exported. Notes data stored in the MDS repository is exported too. When set to `0`, this data is not exported. This argument defaults to `0`.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.22.1.3 Examples

The following example exports a Spaces application and all possible data to a file named `myExport.ear`.

```
wls:/weblogic/serverConfig> exportWebCenterApplication(appName='webcenter',
fileName='myExport.ear', exportCustomizations=1, exportData=1)
```

The following example exports a test application. In this case, data created during testing (such as lists, space events, links, tags, and so on) is not required.

```
wls:/weblogic/serverConfig> exportWebCenterApplication(appName='webcenter',
fileName='export.ear')
```

10.22.2 importWebCenterApplication

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.2.1 Description

(Spaces application only) Imports a Spaces application from an export archive file to a server.

After importing the Spaces application you will need to restart the managed server where the application is deployed.

10.22.2.2 Syntax

```
importWebCenterApplication(appName, fileName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which to perform this operation.
fileName	Name of the export archive that you want to import.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.22.2.3 Example

The following example imports a Spaces application from the export archive `myExport.ear`.

```
wls:/weblogic/serverConfig> importWebCenterApplication(appName='webcenter',
fileName='myExport.ear')
```

10.22.3 exportGroupSpaces

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.3.1 Description

(Spaces application only) Exports one or more named spaces to an export archive (.EAR), using the filename specified.

Space-related data*, application customizations, and security information is included in the export archive.

*Only internal space-related data stored in the Spaces database is exported. For example, data associated with WebCenter Portal services such as Activity Streams, Events, Feedback, Lists, Links, Message Boards, People Connections, Profiles, and Tags.

You must take the Spaces offline, even if only temporarily, to prevent data conflicts during the export process.

Note: You cannot use this command to export the Home space.

10.22.3.2 Syntax

```
exportGroupSpaces(appName, fileName, names, [forceOffline, exportContentDirectory,
server, applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which to perform this operation—always `webcenter`.
fileName	Name of the local file to which the export will be written.

Argument	Definition
names	Names of the spaces that you want to export. Separate multiple space names with a comma. For example: names='sales, finance' **Note:** Do not enter display names here. You must enter the space name that is specified in the space URL. The space name is available from the About Space dialog.
forceOffline	Optional. Specifies whether to take the spaces offline before starting export process. Valid values are `1` and `0`. ■ `1` takes the spaces offline before starting the export process. ■ `0` attempts to export the spaces. If one or more spaces are currently online, an information message requests that you take the spaces offline. The defaults is 0.
exportContentDirectory	Optional. Indicates whether to export content directory files for portal resources. Valid values are `1` and `0`. ■ `1` Exports the entire content directory associated with portal resources ■ `0` Excludes all portal resource content directories. If excluded, you can manually migrate files for portal resources used by the selected spaces. The default is 0. **Note:** The Spaces application stores portal resource content under a single 'shared' content directory (`oracle\webcenter\siteresources\scopedMD\shared`).The entire directory is exported; you cannot import content specific to selected spaces.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.22.3.3 Example

The following example exports two spaces (`mySpace1` and `mySpace2`) from Spaces application named `webcenter`.

```
wls:/weblogic/serverConfig> exportGroupSpaces(appName='webcenter',
fileName='myExport.ear', names='mySpace1, mySpace2')
```

The following example takes `mySpace1` and `mySpace2` offline and then exports both spaces, together with content associated with their portal resources to and archive named `myExport.ear`:

```
wls:/weblogic/serverConfig> exportGroupSpaces(appName='webcenter',
fileName='myExport.ear', names='mySpace1, mySpace2', forceOffline=1,
exportContentDirectory=1)
```

10.22.4 exportGroupSpaceTemplates

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.4.1 Description

(Spaces application only) Exports one or more space templates to an export archive (.EAR), using the filename specified.

10.22.4.2 Syntax

```
exportGroupSpaceTemplates(appName, fileName, names, [exportContentDirectory,
server, applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which to perform this operation—always `webcenter`.
fileName	Name of the local file to which the export will be written.
names	Names of the space templates that you want to export. Separate multiple template names with a comma.
exportContentDirectory	Optional. Indicates whether to export content directory files for portal resources. Valid values are `1` and `0`. ■ `1` Exports the entire content directory associated with portal resources ■ `0` Excludes all portal resource content directories. If excluded, you can manually migrate files for portal resources used by the selected space templates. The default is `0`. **Note:** The Spaces application stores portal resource content under a single 'shared' content directory (`oracle\webcenter\siteresources\scopedMD\shared`).The entire directory is exported; you cannot import content specific to selected space templates.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.22.4.3 Example

The following example exports two space templates (`mySpaceTemplate1` and `mySpaceTemplate2`), together with content associated with their portal resources, to an archive named `myExport.ear`:

```
wls:/weblogic/serverConfig> exportGroupSpaceTemplates(appName='webcenter',
fileName='myExport.ear', names='mySpaceTemplate1, mySpaceTemplate2',
exportContentDorectory=1)
```

10.22.5 importGroupSpaces

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.5.1 Description

(Spaces application only) Imports one or more spaces or space templates from an export archive.

Note: You must take existing spaces offline, even if only temporarily, to prevent data conflicts during the import process.

10.22.5.2 Syntax

```
importGroupSpaces(appName, fileName, [importCustomizations,importSecurity,
importData, parentSpace, forceOffline, overwriteContentDirectory, server,
applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application in which to perform this operation—always `webcenter`.
fileName	Name of the archive file that you want to import.
importCustomizations	Optional. Indicates whether to import space customizations from the export archive. Valid values are 1 and 0. When set to 0: ■ New spaces are imported without customizations (that is, default task flows are imported without any customizations and the default space settings are used). ■ If you are importing a space that already exists on the target, existing customizations on the target are preserved. This argument defaults to 1. Note: Portlet and page customizations are always imported. For information about which customizations are optional on import, read "Understanding Spaces Export and Import" in *Oracle Fusion Middleware Administrator's Guide for Oracle WebCenter*.
importSecurity	Optional. Indicates whether to import space member details from the export archive. Valid values are 1 and 0. When set to 1, roles and permissions for the space, as well as member details and their role assignments are imported. When set to 0, only the roles and permissions are imported. This option is useful when migrating between stage and production environments and where member details, added during the testing phase, are no longer required. This argument defaults to 1.
importData	Optional. Indicates whether to import data from the export archive. Valid values are 1 and 0. When set to `1`, space-related data stored in the Spaces database for various WebCenter Portal services (Activity Streams, Events, Feedback, Lists, Links, Message Boards, People Connections, Profiles, and Tags) is imported. When set to `0`, this data is not imported. This option is useful when migrating between stage and production environments and where test data is no longer required. This argument defaults to `1`.
forceOffline	Optional. Takes the space(s) offline before import. Valid values are 1 and 0. When set to 1, all space(s) are taken offline. This argument defaults to 0.

Argument	Definition
parentSpace	Optional. Name of the parent space under which to place spaces in the archive. If specified, imported spaces become children of the parent space. This argument defaults to null. When no parent is specified, archived spaces are imported as root spaces. Note: If the archive contains space templates, this argument is ignored.
overwriteContentDirectory	Optional. Indicates whether to overwrite existing content directory files (used by portal resources) in the target with the files in the archive. Valid values are `1` and `0`. ■ `1` Overwrites the content directory files in the target with the files in the archive ■ `0` Only imports new files, that is, files that do not exist in the target content directory. The default is 0. **Note:** The Spaces application stores portal resource content under a single 'shared' content directory (`oracle\webcenter\siteresources\scopedMD\shared`).The entire directory is exported; you cannot import content specific to selected space templates.
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.22.5.3 Example

The following example imports spaces or space templates from an archive named `myExport.ear` to a Spaces application named `webcenter`.

```
wls:/weblogic/serverConfig> importGroupSpaces(appName='webcenter',
fileName='myExport.ear')
```

The following example takes all existing spaces in the target in `myExport.ear` offline and then imports all the spaces in `myExport.ear` under the "Sales" space. Space customizations, together with content associated with portal resources are imported to the target. Test data and security details are not required:

```
wls:/weblogic/serverConfig> importGroupSpaces(appName='webcenter',
fileName='myExport.ear', importCustomizations=1, importSecurity=0, importData=0,
parentSpace="Sales", forceOffline=1, overwriteContentDirectory=1)
```

10.22.6 exportWebCenterResource

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.6.1 Description

Exports a single portal resource to an export archive (.EAR), using the filename specified.

10.22.6.2 Syntax

```
exportWebCenterResource(appName, fileName, resourceType, resourceGUID,
[spaceName, exportContentDirectory, server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
fileName	Name of the local file to which the export will be written.
resourceType	Type of resource to export. Valid values include: `pageTemplate`, `contentPresenter`, `pageStyle`, `navigation`, `resourceCatalog`, `skin`, `taskFlow`, `mashupStyle`.
resourceGUID	Unique ID (GUID) of the resource to export.
spaceName	Optional. (Spaces application only) Name of the space containing the resource to export. Use this argument to export resources that are owned by a particular Space. Omit this argument if you want to export application-level resources for the Spaces application or to export resources for a WebCenter Portal application. This argument defaults to null (application-level resources exported).
exportContentDirectory	Optional. Indicates whether to export content directories associated with this portal resource. Valid values are `1` and `0`. ■ `1` Exports content referenced by the portal resource. Entire directories are exported rather than individual files. If the portal resource uses content from multiple directory locations, all directories are exported. For example, if a skin references two files (`...\shared\skins\logos\mylogo.gif` and `...\shared\skins\icons\myicon.gif`), the entire content of `\logos` and `\icons` are exported. ■ `0` Excludes content referenced by the resource. If excluded, you can manually migrate the content directory/individual files that this resource uses. The default is 0.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.22.6.3 Example

The following example exports a page template from `MySpace` to a local file named `myPageTemplateExport.ear`:

```
wls:/weblogic/serverConfig> exportWebCenterResource(appName='webcenter',
fileName='myPageTemplateExport.ear', resourceType='pageTemplate',
resourceGUID='gsr47d9a5ac_7398_439a_97d2_8b54ce905f7e, spaceName='MySpace')
```

The following example exports a skin from a WebCenter Portal application named `myPortalApp` to a local file named `mySkinExport.ear`. Content directories referenced by the skin are exported too:

```
wls:/weblogic/serverConfig> exportWebCenterResource(appName='myPortalApp',
```

```
fileName='mySkinExport.ear', resourceType='skin',
resourceGUID='gsr47d9a5ac_7398_439a_97d2_8b54ce905f7e, exportContentDirectory=1)
```

10.22.7 importWebCenterResource

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.7.1 Description

Imports a single portal resource from an export archive (.EAR), using the filename specified.

10.22.7.2 Syntax

```
importWebCenterResource(appName, fileName, resourceType, [spaceName, server,
applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation.
fileName	Name of the archive file that you want to import.
resourceType	Type of resource to import. Valid values include: `pageTemplate`, `contentPresenter`, `pageStyle`, `navigation`, `resourceCatalog`, `skin`, `taskFlow`, `mashupStyle`.
spaceName	Optional. (Spaces application only) Name of the space into which the resource is to be imported. Omit this argument if you want to import application-level resources for the Spaces application or to import resources for a WebCenter Portal application. This argument defaults to null (application-level resources imported).
overwriteContentDirectory	Optional. Indicates whether to overwrite existing content directories (used by the portal resource) in the target with the files in the archive. Valid values are `1` and `0`. ■ `1` Overwrites resource content directory files in the target with the files in the archive ■ `0` Only imports new files, that is, files that do not exist in the target content directory. The default is 0.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.22.7.3 Example

The following example imports a page template from an archive named `myPageTemplateExport.ear` to `MySpace` in the Spaces application:

```
wls:/weblogic/serverConfig> importWebCenterResource(appName='webcenter',
fileName='myPageTemplateExport.ear', spaceName='MySpace',
resourceType='pageTemplate')
```

The following example imports a skin from an archive named `mySkinExport.ear` to a WebCenter Portal application named `myPortalApp`. On import, content directories referenced by the skin are overwritten:

```
wls:/weblogic/serverConfig> importWebCenterResource(appName='myPortalApp',
fileName='mySkinExport.ear', resourceType='skin', overwriteContentDirectory=1)
```

10.22.8 exportPortletClientMetadata

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.8.1 Description

Exports portlet client metadata and producer customizations and personalizations, for a Framework application. This command exports metadata for *all* the application's producers to a named export archive (.EAR file). You cannot opt to export metadata for specific producers.

Only use this command to migrate producer data associated with WebCenter Portal applications developed using WebCenter Portal: Framework inJDeveloper. Do not use this command for the Spaces application.

10.22.8.2 Syntax

```
exportPortletClientMetadata(appName, fileName, [exportPersonalizations, server,
applicationVersion])
```

Argument	Definition
appName	Name of the Framework application in which to perform this operation.
fileName	Name of the export archive (.EAR) to which you want the export to be written.
exportPersonalizations	Optional. Valid values are `1` (true) and `0` (false). When set to `1`, personalizations for *all* producers are exported. When set to `0`, personalizations are not exported. This argument defaults to `1`.
server	Optional. Name of the managed server where the Framework application is deployed. For example, `WC_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 Framework application is deployed.

10.22.8.3 Example

The following example exports portlet client metadata and producer customizations to an export archive named `myExport.ear`. Personalizations are not exported.

```
wls:/weblogic/serverConfig> exportPortletClientMetadata(appName='myApp',
fileName='myExport.ear', exportPersonalizations=0)
```

10.22.9 importPortletClientMetadata

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.9.1 Description

Imports portlet client metadata and producer customizations and personalizations from a named export archive.

Producer personalizations are optional on export. Producer personalizations are imported if the export archive specified includes personalizations.

Only use this command to migrate producer data for a WebCenter Portal application developed using WebCenter Portal: Framework in JDeveloper. Do not use this command for the Spaces application.

10.22.9.2 Syntax

```
importPortletClientMetadata(appName, fileName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the Framework application in which to perform this operation.
fileName	Name of the export archive that you want to import.
server	Optional. Name of the managed server where the Framework application is deployed. For example, `WC_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 Framework application is deployed.

10.22.9.3 Example

The following example imports portlet client metadata and producer customizations and personalizations from a WebCenter export archive named `myExport.ear`.

```
wls:/weblogic/serverConfig> importPortletClientMetadata(appName='app1',
fileName='myExport.ear')
```

10.22.10 importWebCenterTranslations

Module: Oracle WebCenter

Use with WLST: Online

10.22.10.1 Description

Spaces application only. Imports translated content (XLF files) to MDS and the WebCenter repository for use in the Spaces application.

10.22.10.2 Syntax

```
importWebCenterTranslations(appName, server, mdsRootDir, [applicationVersion])
```

Argument	Definition
appName	Name of the Spaces application application in which to perform this operation—always `webcenter`.
server	Name of the target managed server on which the Spaces application is deployed. For example, `WC_Spaces`.
mdsRootDir	MDS root directory on the file system that contains translated XLF files.

Argument	Definition
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter application is deployed.

10.22.10.3 Example

The following example imports translated content in the directory `/scratch/shared/newmd` to MDS and the WebCenter Portal repository:

```
wls:/weblogic/serverConfig> importWebCenterTranslations(appName='webcenter',
server='WC_Spaces', mdsRootDir='/scratch/shared/newmd')
```

10.22.11 setSpaceState

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.11.1 Description

(Spaces application only) Takes a space offline or brings a space online.

10.22.11.2 Syntax

```
setSpaceState(appName, spaceName, offline, [server, applicationVersion])
```

Argument	Definition
appName	Name of the application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
spaceName	Name of the space you want to take offline or bring online.
offline	Specifies whether to take the space offline or bring it back online. Valid values are `1` and `0`: ■ 1 takes the space offline ■ 0 brings the space online
server	Optional. Name of the managed server where the Spaces application is deployed. For example, `WC_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 Spaces application is deployed.

10.22.11.3 Example

The following example takes `MySpace` offline:

```
wls:/weblogic/serverConfig> setSpaceState(appName='webcenter',
spaceName='MySpace', offline=1)
```

10.22.12 showProducerImportFailures

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.12.1 Description

Lists outstanding producer imports for a named WebCenter Portal application.

Producer import fails if a producer used by the application is not available when the application first starts after deployment.

10.22.12.2 Syntax

```
showProducerImportFailures(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
server	Name of the managed server on which the application is deployed.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter Portal application is deployed.

10.22.12.3 Example

The following example shows import failures for an application named `webcenter`:

```
wls:/weblogic/serverConfig> showProducerImportFailures(appName='webcenter')
```

10.22.13 retryAllFailedProducerImports

Module: Oracle WebCenter Portal

Use with WLST: Online

10.22.13.1 Description

Imports outstanding producer metadata.

Producer import can fail if a producer used by the application is not available when the application first starts after deployment. Use this command to import metadata for any producers for which metadata import previously failed.

10.22.13.2 Syntax

```
retryAllFailedProducerImports(appName, [server, applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
server	Name of the managed server on which the WebCenter Portal application is deployed.
applicationVersion	Optional. Version number of the deployed application. Required if more than one version of the WebCenter Portal application is deployed.

10.22.13.3 Example

The following example imports missing producer metadata for an application named `webcenter`:

```
wls:/weblogic/serverConfig> retryAllFailedProducerImports(appName='webcenter')
```

10.23 WebCenter Portal Upgrade

Use the commands listed in Table 10–30 when upgrading from a previous WebCenter Portal release.

See also, *Oracle Fusion Middleware Upgrade Guide for Oracle SOA Suite, WebCenter, and ADF*.

Table 10–30 WebCenter Portal Upgrade WLST Commands

Use this command...	To...	Use with WLST...
upgradeWebCenterDomain	Upgrade a WebCenter Portal domain.	Offline
upgradeWebCenterPermissions	Upgrade WebCenter Portal permissions.	Online
upgradeWebCenterApplication	Upgrade a WebCenter Portal application.	Online

10.23.1 upgradeWebCenterDomain

Module: Oracle WebCenter Portal

Use with WLST: Offline

10.23.1.1 Description

Upgrades a WebCenter Portal Domain from 11.1.1.2.0 or 11.1.1.3.0 to 11.1.1.4.0

10.23.1.2 Syntax

```
upgradeWebCenterDomain(domainDirName, [oracleHome], [upgradeCustomSpaces])
```

Argument	Definition
domainDirName	Full path to the domain's home directory. For example, `/home/Oracle/Domains/wc_domain`.
oracleHome	Optional. Path to WebCenter Portal's Oracle home directory. For example, `/home/Oracle/Middleware/Oracle_WC`.
upgradeCustomSpaces	Optional. Determines whether to upgrade the `custom.webcenter.spaces` shared library. Valid values are `1` (true) and `0` (false). Set to `1` if you customized the Spaces application and you want your customizations to be included when you upgrade. The default value is `0`.

10.23.1.3 Example

The following example upgrades a WebCenter Portal domain named `base_domain`:

```
wls:/weblogic/serverConfig> upgradeWebCenterDomain(domainDirName='/mw_home/user_
project/domains/base_domain');
```

10.23.2 upgradeWebCenterPermissions

Module: Oracle WebCenter Portal

Use with WLST: Online

10.23.2.1 Description

Upgrades permissions for the Spaces application.

This command creates additional application roles and grants some additional permissions that are requirement for Spaces 11.1.1.4.0.

10.23.2.2 Syntax

```
upgradeWebCenterPermissions()
```

10.23.2.3 Example

The following example upgrades permissions for the Spaces application:

```
wls:/weblogic/serverConfig> upgradeWebCenterPermissions();
```

10.23.3 upgradeWebCenterApplication

Module: Oracle WebCenter Portal

Use with WLST: Online

10.23.3.1 Description

Upgrades a Spaces application from 11.1.1.2.0 or 11.1.1.3.0 to 11.1.1.4.0.

10.23.3.2 Syntax

```
upgradeWebCenterApplication(appName, [server], [applicationVersion])
```

Argument	Definition
appName	Name of the WebCenter Portal application in which to perform this operation. For the Spaces application, the name is always `webcenter`.
server	Optional. Name of the managed server where the WebCenter Portal application is deployed. For example, `WC_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 Portal application is deployed.

10.23.3.3 Example

The following example upgrades the Spaces application:

```
wls:/weblogic/serverConfig> upgradeWebCenterApplication(appName='webcenter');
```

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 WLST Command Categories"
- Section 11.2, "ADF-Specific WLST Commands"

> **Note:** To use these ADF custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide*.

11.1 Overview of WLST Command Categories

Use the ADF-based URL Connections WLST commands to navigate the hierarchy of configuration or runtime beans and control the prompt display. Use the `getADFMArchiveConfig` command to manage the `ADFMArchiveConfig` object.

11.2 ADF-Specific WLST Commands

Use the commands in Table 11–1 to managing URL-based connections.

Table 11–1 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_listUrlConnection	List a new URL connection.	Online or Offline
getADFMArchiveConfig	Returns a handle to the `ADFMArchiveConfig` object for the specified archive.	Online or Offline

11.2.1 adf_createFileUrlConnection

Use with WLST: Online or Offline

11.2.1.1 Description

Use this command to create 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 list the connections of the application.

11.2.4.2 Syntax

```
adf_listURLConnection(appname)
```

Argument	Definition
appname	Application name

11.2.4.3 Example

```
adf_listURLConnection ('myapp')
```

11.2.5 getADFMArchiveConfig

Use with WLST: Online or Offline.

11.2.5.1 Description

Returns a handle to the `ADFMArchiveConfig` object for the specified archive. The returned `ADFMArchiveConfig` object's methods can be used to change application configuration in an archive.

The `ADFMArchiveConfig` object provides the following methods:

- `setDatabaseJboSQLBuilder([value])`—Sets the Database `jbo.SQLBuilder` attribute.

- `getDatabaseJboSQLBuilder()`—Returns the current value of the `jbo.SQLBuilder` attribute.
- `setDatabaseJboSQLBuilderClass([value])`—Sets the Database `jbo.SQLBuilderClass` attribute. Value is the full name of the custom builder class.
- `getDatabaseJboSQLBuilderClass()`—Returns the current value of the `jbo.SQLBuilderClass` attribute.
- `setDefaultRowLimit([value])`—Sets the defaults `rowLimit` attribute. Value is a long specifying the row limit (Default -1).
- `getDefaultRowLimit()`—Returns the current value of the `rowLimit` attribute.
- `save([toLocation])`—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.

11.2.5.2 Syntax

```
archiveConfigObject = ADFMAdmin.getADFMArchiveConfig(fromLocation)
```

Argument	Definition
fromLocation	The name of the ear file, including its complete path.

The syntax for `setDatabaseJboSQLBuilder([value])` is:

```
archiveConfigObject.setDatabaseJboSQLBuilder([value])
```

Argument	Definition
value	The value of the `jbo.SQLBuilder` attribute. Valid values are: 'Oracle' (Default), 'OLite', 'DB2', 'SQL92', 'SQLServer', or 'Custom. If 'Custom' is specified, then the `jbo.SQLBuilderClass` attribute should also be set.

The syntax for `getDatabaseJboSQLBuilder()` is:

```
archiveConfigObject.getDatabaseJboSQLBuilder()
```

The syntax for `setDatabaseJboSQLBuilderClass([value])` is:

```
archiveConfigObject.setDatabaseJboSQLBuilderClass([value])
```

Argument	Definition
value	The value of the `jbo.SQLBuilderClass` attribute.

The syntax for `getDatabaseJboSQLBuilderClass)` is:

```
archiveConfigObject.getDatabaseJboSQLBuilderClass()
```

The syntax for `setDefaultRowLimit([value])` is:

```
archiveConfigObject.setDefaultRowLimit([value])
```

Argument	Definition
value	The value of the `rowLimit` attribute.

The syntax for `getDefaultRowLimit()` is:

```
archiveConfigObject.getDefaultRowLimit([value])
```

The syntax for `save([toLocation])` is:

```
archiveConfigObject.save([toLocation])
```

Argument	Definition
toLocation	The file name along with the absolute path to store the changes.

11.2.5.3 Example

In the following example. the `jbo.SQLBuilder` attribute is set to 'DB2'.

```
wls:/offline> archive =
              ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDatabaseJboSQLBuilder(value='DB2')
wls:/offline> archive.save()
```

In the following example, the `jbo.SQLBuilder` attribute is removed so that application default is used.

```
wls:/offline> archive =
              ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDatabaseJboSQLBuilder()
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

In the following example, the `jbo.SQLBuilder` attribute is set to 'Custom', and the `jbo.SQLBuilderClass` attribute is set to the class 'com.example.CustomBuilder'.

```
wls:/offline> archive =
              ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDatabaseJboSQLBuilder('Custom')
wls:/offline> archive.setDatabaseJboSQLBuilderClass('com.example.CustomBuilder')
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

In the following example, the `rowLimit` attribute is set to 100.

```
wls:/offline> archive = getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDefaultRowLimit(100)
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

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
Database Access Descriptor Commands	Create, edit, or delete a general DAD or Portal DAD.
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 Java EE 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 JRF custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Table 13–1 JRF Commands

Use this command...	To...	Use with WLST...
applyJRF	Configures a Managed Server or cluster with Java Required Files applications and services.	Online or Offline
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')
```

14

Web Services Custom WLST Commands

The following sections describe the WebLogic Scripting Tool (WLST) commands for Oracle Fusion Middleware Infrastructure Web services, which includes SOA composites, ADF Business Components, and WebCenter services. You can use these commands to manage Web services from the command line.

Topics in this chapter include:

For additional details about using these WLST commands for Web services, see the *Security and Administrator's Guide for Web Services*.

Notes: To use the Web Services custom WLST commands, you must invoke WLST from the Oracle Common home directory. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

To display the help for the Web service and client management, policy management, and deployment descriptor migration commands, connect to a running instance of the server and enter `help('WebServices')`.

To display the help for the policy set management and Oracle WSM repository management commands, connect to a running instance of the server and enter `help('wsmManage')`.

14.1 Overview of Web Services WLST Commands

You can use the Web services WLST commands, in online mode, to:

- Perform Web service configuration and Oracle WSM policy management tasks
- Upgrade the Oracle WSM Repository with new predefined policies with each release

- Migrate post-deployment policy changes persisted in proprietary deployment descriptor (PDD) files for ADF Business Components and WebCenter services and propagate policy changes to all server instances in a domain.

The Web services WLST commands manage deployed, active, and running Web services applications. They can be executed everywhere in WLST online mode, for example:

```
wls:/domain/serverConfig
wls:/domain/domainRuntime
```

The Web services WLST configuration and policy management commands perform many of the same management functions that you can complete using Fusion Middleware Control. When using the WLST commands to manage a Web service of an ADF or WebCenter application, you can apply the change only to a Web service deployed in an application on a specific server. If the application is deployed in a cluster or multi-server environment, you need to make the same change to each of the servers to which the application is deployed. Additionally, when you set or change an attached policy in ADF and WebCenter Web service and client applications, you must restart the application for the changes to take effect.

In contrast, if you are using the WLST commands to manage a SOA composite, you only need to issue the command once, and the change is propagated to all the server instances in the composite. When you set or change an attached policy in a SOA composite, you do not need to restart it. The SOA fabric runtime engine internally implements all of the policy management changes.

14.1.1 Specifying Application, Composite, and Service Names

The Web service WLST commands configure a Web service for a specific application. Therefore, the application path name has to uniquely identify the application and the server instance to which it is deployed.

Specifying a Web Service Application Name

To specify a Web service application in a WLST command, use the following format:

```
[/domain/server/]application[#version_number]
```

Parameters shown in brackets `[]` are optional. The following examples show the sample format for a Web service application name:

```
/soainfra/AdminServer/HelloWorld#1_0
/soainfra/server1/HelloWorld#1_0
```

If there is only one deployed instance of an application in a domain, you may omit the `domain/server` parameter, as shown in the following example:

```
HelloWorld#1_0
```

In all other instances, the `domain/server` parameter is required. If it is not specified and WLST finds more than one deployment of the same application on different servers in the domain, you are prompted to specify the domain and the server names.

Oracle Infrastructure Web Services client applications are deployed directly to WebLogic Server server instances. Each client application is managed separately. For example, if the application `myapp` is deployed to both the `AdminServer` and `server1` instances in the domain `mydomain`, then you need to issue configuration commands to each of the servers using the appropriate application path name:

```
/mydomain/AdminServer/myapp#1_0
```

```
/mydomain/server1/myapp#1_0
```

Specifying a SOA Composite Name

When there are multiple SOA partition folders in a domain, you must specify the partition name and the composite name using the following format:

```
partition/composite[version]
```

The following example shows the sample format for a SOA composite application name:

```
default/myComposite[1.0]
```

If there is a single SOA server (non-clustered) and only one SOA partition folder in a domain, you may omit the `partition` parameter, as shown in the following example:

```
myComposite[1.0]
```

Specifying a Service Name

When there are multiple versions (namespaces) of a Web service name, you must specify the namespace and the service name using the following format:

```
{http://namespace/}serviceName
```

Note the following:

- For Web service and client management commands, and policy management commands, you do not need to enter the namespace if there is only one service name qualified. If there are multiple versions of the service and you do not specify the namespace with the service name, an exception is thrown.
- For policy set management commands, both the namespace and service name are required.

14.1.2 Web Services WLST Command Categories

Web services WLST commands are divided into the categories described in Table 14–1.

Table 14–1 Web Services WLST Command Categories

Command Category	Definition
Section 14.2, "Web Service and Client Management Commands"	View and manage Web services for the service and client.
Section 14.3, "Policy Management Commands"	View and manage directly-attached policies for the service and client.
Section 14.4, "Policy Set Management Commands"	View and manage globally-available policy sets within repository sessions.
Section 14.5, "Oracle WSM Repository Management Commands"	Manage the Oracle WSM repository with new predefined policies provided in the latest installation of the software, as well as import and export documents into and from the repository.
Section 14.6, "Deployment Descriptor Migration Commands"	Migrate proprietary deployment descriptors for scaling post-deployment policy configuration changes in a cluster or propagating the changes to all server instances of the application in the domain.

14.2 Web Service and Client Management Commands

Use the WLST commands listed in Table 14–2 to view and manage Web services for deployed, active, and running Web service applications.

Table 14–2 Web Service and Client Management WLST Commands

Use this command...	To...	Use with WLST...
listWebServices	List the Web service information for an application, composite, or domain.	Online
listWebServicePorts	List the Web service ports for a Web service application or SOA composite.	Online
listWebServiceConfiguration	List Web services and port configuration for an application or SOA composite.	Online
setWebServiceConfiguration	Set or change the Web service port configuration for a Web service application or SOA composite.	Online
listWebServiceClients	List Web service client information for an application, SOA composite, or domain.	Online
listWebServiceClientPorts	List Web service client ports information for an application or SOA composite.	Online
listWebServiceClientStubProperties	List Web service client port stub properties for an application or SOA composite.	Online
setWebServiceClientStubProperty	Set, change, or delete a single stub property of a Web service client port for an application or SOA composite.	Online
setWebServiceClientStubProperties	Configure the set of stub properties of a Web service client port for an application or SOA composite.	Online

14.2.1 listWebServices

Command Category: Web Service and Client Management

Use with WLST: Online

14.2.1.1 Description

Lists the Web service information for an application, SOA composite, or domain. If you don't specify a Web service application or a SOA composite, the command lists all services in all applications and composites for every server instance in the domain.

You can specify the amount of information to be displayed in the output using the `detail` argument. When specified, the output provides endpoint (port) and policy details for all applications and composites in the domain, the secure status of the endpoints, any configuration overrides and constraints, and if the endpoints have a valid configuration. A subject is considered secure if the policies attached to it (either directly or globally) enforce authentication, authorization, or message protection behaviors. Because you can specify the priority of a global or directly attached policy (using the `reference.priority` configuration override), the `effective` field indicates if the directly attached policies are in effect for the endpoint.

Note that to simplify endpoint management, all directly attached policies are shown in the output regardless of whether they are in effect. In contrast, only globally attached policies that are in effect for the endpoint are displayed. For more information, see "How the Effective Set of Policies is Calculated" in *Security and Administrator's Guide for Web Services*.

The output is listed by each application deployed as shown in the following example:

```
/domain/server/application#version_number:
     moduleName=helloModule, moduleType=web,
serviceName={http://namespace/}service
/soainfra/AdminServer/soa-infra:
     compositeName=default/HelloWorld[1.0],  moduleType=soa, serviceName=service
```

Notes: The `listWebServices` command output does not include details on SOA components, including policy attachments.

For applications assembled prior to PS5, the namespace is not displayed with the `serviceName` in the output.

14.2.1.2 Syntax

```
listWebServices (application,composite,[detail])
```

Argument	Definition
application	Name and path of the application for which you want to list the Web services. For example, `/domain/server/application#version_number` If specified, all Web services in the application are listed.
composite	Name of the SOA composite for which you want to list the Web services. For example, `default/HelloWorld[1.0]` If specified, all Web services in the composite are listed.
detail	Optional. Specifies whether to list port and policy details for the Web service. Valid values are: ■ `true`—Output includes details about the service, the port, and the policies. ■ `false`—Output lists only the services. The default is `false`.

14.2.1.3 Examples

The following example lists all the Web services in all applications and composites in the domain. Sample output is shown in this example.

```
wls:/soainfra/serverConfig> listWebServices()
/soainfra/AdminServer/soa-infra :
     compositeName=default/HelloWorld[1.0], moduleType=soa, serviceName=service
     compositeName=default/Project1[1.0], moduleType=soa,
serviceName=bpelprocess1_client_ep

/soainfra/AdminServer/HelloWorld#1_0 :
     moduleName=j2wbasicPolicy, moduleType=web, serviceName=WssUsernameService
```

The following example sets the `detail` argument to `true`. Sample output is shown in this example. Note that the directly attached policy is not in effect for the endpoint TestPort in the application jaxws-sut.

```
wls:/jrfServer_domain/serverConfig> listWebServices(detail='true')

/jrfServer_domain/jrfServer_admin/jaxws-sut-no-policy :
        moduleName=jaxws-service, moduleType=web,
serviceName={http://namespace/}TestService
        enableTestPage: true
        enableWSDL: true

                TestPort
http://host.us.oracle.com:9315/jaxws-service/TestService
                enable: true
                enableREST: false
                enableSOAP: true
                maxRequestSize: -1
                loggingLevel: NULL
                wsat.flowOption: NEVER
                wsat.version: DEFAULT
                Constraint: No Constraint
                        (global) security : oracle/wss_saml_or_username_token_
service_policy, enabled=true

/policysets/global/all-domains-default-web-service-policies : Domain("*")
                                        reference.priority=1
                Constraint: HTTPHeader('VIRTUAL_HOST_TYPE','external')
                        (global) security : oracle/wss10_message_protection_
service_policy, enabled=true
                                /policysets/global/domainExternal : Domain("*")
                Attached policy or policies are valid; endpoint is secure.


/jrfServer_domain/jrfServer_admin/jaxws-sut :
        moduleName=jaxws-sut-service, moduleType=web,
serviceName={http://namespace/}TestService
        enableTestPage: true
        enableWSDL: true

                TestPort
http://host.us.oracle.com:9315/jaxws-sut-service/TestService
                enable: true
                enableREST: false
                enableSOAP: true
                maxRequestSize: -1
                loggingLevel: NULL
                wsat.flowOption: NEVER
                wsat.version: DEFAULT
                management : oracle/log_policy, enabled=true
                security : oracle/wss_username_token_service_policy , enabled=true
, effective=false
                Constraint: No Constraint
                        (global) security : oracle/wss_saml_or_username_token_
service_policy, enabled=true

/policysets/global/all-domains-default-web-service-policies : Domain("*")
                                        reference.priority=1
                Constraint: HTTPHeader('VIRTUAL_HOST_TYPE','external')
                        (global) security : oracle/wss10_message_protection_
```

```
service_policy, enabled=true
                                /policysets/global/domainExternal : Domain("*")
               Attached policy or policies are valid; endpoint is secure.
```

14.2.2 listWebServicePorts

Command Category: Web Service and Client Management

Use with WLST: Online

14.2.2.1 Description

List the Web service port names and the endpoint URLs for a Web service application or SOA composite.

The output will display the port name and endpoint URL of the Web service port. For example:

```
JRFWssUsernamePort         http://localhost:7001/j2wbasicPolicy/WssUsername
```

14.2.2.2 Syntax

```
listWebServicePorts(application,moduleOrCompName,moduleType,serviceName)
```

Argument	Definition
application	Name and path of the application for which you want to list the Web services port information. For example, `/domain/server/application#version_number` To list the port information for an application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to list the Web services port information. To list the port information for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with Web modules (including EJB Web services.) ■ `soa`—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite for which you want to list the port information. For example, `{http://namespace/}serviceName`

14.2.2.3 Example

The following example lists the Web service ports and endpoint URLs for the `j2wbasicPolicy` service in the `soainfra/AdminServer/HelloWorld#1_0` application. Note that the `WssUsernameService` module name is specified, and the `moduleType` is set to `web`.

```
wls:/soainfra/serverConfig> listWebServicePorts
( '/soainfra/AdminServer/HelloWorld#1_0',
'WssUsernameService','web','{http://namespace/}j2wbasicPolicy')

JRFWssUsernamePort      http://localhost:7001/j2wbasicPolicy/WssUsername
```

14.2.3 listWebServiceConfiguration

Command Category: Web Service and Client Management

Use with WLST: Online

14.2.3.1 Description

List the Web service port configuration for a Web service application or SOA composite.

The output will display the configuration information for the Web service port. For example:

```
enableREST: false
maxRequestSize: -1
```

14.2.3.2 Syntax

```
listWebServiceConfiguration(application,moduleOrCompName,moduleType,serviceName,
[subjectName])
```

Argument	Definition
application	Name and path of the application for which you want to list the Web services port configuration. For example, `/domain/server/application#version_number` To list the port configuration for a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to list the Web services port configuration. To list the port configuration for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with Web modules (including EJB Web services.). ■ `soa`—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite for which you want to list the port configuration. For example, `{http://namespace/}serviceName`
subjectName	Optional. Policy subject, port, or operation for which you want to list configuration information.

14.2.3.3 Example

The following example lists the Web service and port configuration information for the application `HelloWorld#1_0` for the server `soa1` in the domain `soainfra`. In this example, the Web module name is `j2wbasicPolicy`, the service name is `WssUsernameService`, and the subject is a port named `JRFWssUsernamePort`.

```
wls:/wls-domain/serverConfig>listWebServiceConfiguration
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort')
```

14.2.4 setWebServiceConfiguration

Command Category: Web Service and Client Management

Use with WLST: Online

14.2.4.1 Description

Set or change the Web service port configuration for a Web service application or SOA composite.

Additional information about using this command is provided in "Configuring the Web Service Endpoint" in *Security and Administrator's Guide for Web Services*.

14.2.4.2 Syntax

```
setWebServiceConfiguration(application,moduleOrCompName,moduleType,serviceName,
subjectName,itemProperties)
```

Argument	Definition
application	Name and path of the application for which you want to set or change the Web services port configuration. For example, `/domain/server/application#version_number` To set or change the port configuration for a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to set or change the Web services port configuration. To set or change the port configuration for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with Web modules (including EJB Web services.). ■ `soa`—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite for which you want to set or change the port configuration. For example, `{http://namespace/}serviceName`
subjectName	Policy subject, port or operation name for which you want to set or change the configuration information.

Argument	Definition
itemProperties	Configurable properties that you can set or change. Specify the properties using the following format: `("name","value")` Valid port configuration name and value pairs are as follows: ■ `enable`—`true` or `false`. Default is `true`. ■ `enableTestPage`—`true` or `false`. Default is `true`. ■ `enableWSDL`—`true` or `false`. Default is `true`. ■ `enableREST`—`true` or `false`. Default is `false`. ■ `maxRequestSize`—long integer, `-1` for values not set. The default is `-1`. ■ `loggingLevel`—`NULL`, `FINEST...SEVERE` (java.util.logging.Level). The default is `NULL`. ■ `wsat.flowOption`—Atomic transaction flow option. Valid values are: `NEVER`—Do not export transaction coordination context. (This is the default.), `SUPPORTS`—Export transaction coordination context if transaction is available, `MANDATORY`—Export transaction coordination context. An exception is thrown if there is no active transaction. This property is valid for SOA services only. ■ `wsat.version`—Atomic transaction version. Valid values are: `WSAT10`, `WSAT11`,`WSAT12`, and `DEFAULT`. This property is valid for SOA services only. For additional information about the atomic transaction properties, see "Configuring Web Services Atomic Transactions" in *Security and Administrator's Guide for Web Services*. **Note:** If any configuration item contains an unrecognized property name or invalid value, this set command is rejected and an error message is displayed.

14.2.4.3 Example

The following example enables the port `JRFWssUsernamePort` for the service `WssUsernameService` in the Web module `j2wbasicPolicy`. The service is in the application `HelloWorld#1_0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>setWebServiceConfiguration
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort',[("enable", "true")])
```

14.2.5 listWebServiceClients

Command Category: Web Service and Client Management

Use with WLST: Online

14.2.5.1 Description

List Web service clients information for an application, SOA composite, or domain. If neither an application nor a composite is specified, the command lists information about all Web service clients in all applications and composites for every server instance in the domain.

You can specify the amount of information to be displayed in the output using the `detail` argument. When specified, the output provides endpoint (port) and policy details for clients in the domain, the secure status of the endpoints, any configuration

overrides and constraints, and if the endpoints have a valid configuration. A subject is considered secure if the policies attached to it (either directly or globally) enforce authentication, authorization, or message protection behaviors. Because you can specify the priority of a global or directly attached policy (using the `reference.priority` configuration override), the `effective` field indicates if the directly attached policies are in effect for the endpoint.

Note that to simplify endpoint management, all directly attached policies are shown in the output regardless of whether they are in effect. In contrast, only globally attached policies that are in effect for the endpoint are displayed. For more information, see "How the Effective Set of Policies is Calculated" in *Security and Administrator's Guide for Web Services*.

The output is listed by each application deployed as shown in the following examples:

This example shows the output of an *unsecured* endpoint:

```
/soa_domain/soa_server1/soa-infra :
        compositeName=default/Basic_SOA_Client[1.0], moduleType=soa,
serviceRefName=Service1
                Basic_soa_service_pt
serviceWSDLURI=http://host.us.oracle.com:38001/soa-infra/services/default/Basic_
SOA_service/Basic_soa_service.wsdl
                oracle.webservices.contentTransferEncoding=base64
                oracle.webservices.charsetEncoding=UTF-8
                oracle.webservices.operationStyleProperty=document
                wsat.flowOption=WSDLDriven
                oracle.webservices.soapVersion=soap1.1
                oracle.webservices.chunkSize=4096
                oracle.webservices.session.maintain=false
                oracle.webservices.preemptiveBasicAuth=false

oracle.webservices.encodingStyleProperty=http://schemas.xmlsoap.org/soap/encoding/
                oracle.webservices.donotChunk=true
                No attached policies found; endpoint is not secure.
```

This example shows the output for *secured* endpoints:

```
/soa_domain/soa_server1/AsynchronizedBC_asyncbc :
        moduleName=Asychronized-AsynchronizedBC-context-root, moduleType=web,
serviceRefName=callback
                owsm.qa.server.serviceinterface.AppModule_asyncServiceImpl/_
oracleAsyncResponseClient
              Constraint: No Constraint
                (global) security : oracle/wss_username_token_client_policy,
enabled=true
                        /policysets/global/web_callback_add_1 : Module("*")
                Attached policy or policies are valid; endpoint is secure.


/soa_domain/soa_server1/ADF_DC_4 :
        moduleName=wsdl, moduleType=wsconn, serviceRefName=TestService
                TestPort
serviceWSDLURI=http://host.us.oracle.com:12345/jaxws-sut-service/TestService?wsdl
                security : oracle/wss_username_token_client_policy, enabled=true,
effective=false
              Constraint: No Constraint
                (global) security : oracle/wss11_username_token_with_message_
protection_client_policy, enabled=true
                        /policysets/global/PolicySet-Testport : port('TestPort')
                        reference.priority=1
```

```
               Attached policy or policies are valid; endpoint is secure.

/soa_domain/AdminServer/adf_dc_to_bc :
       moduleName=ADF_BC, moduleType=wsconn, serviceRefName=AppModuleService
               AppModuleServiceSoapHttpPort
serviceWSDLURI=http://host.us.oracle.com:12345/ADF_BC-ADF_
BC-context-root/AppModuleService?wsdl
             Constraint: No Constraint
               (global) security : oracle/wss11_username_token_with_message_
protection_client_policy, enabled=true
                       /policysets/global/web_reference_add_1 : Domain("soa_
domain")
               Attached policy or policies are valid; endpoint is secure.
```

14.2.5.2 Syntax

```
listWebServiceClients(application,composite,[detail])
```

Argument	Definition
application	Name and path of the application for which you want to list the Web service clients. For example, `/domain/server/application#version_number` If specified, all Web services clients in the application are listed.
composite	Name of the SOA composite for which you want to list the Web service clients. For example, `default/HelloWorld[1.0]` If specified, all Web service clients in the composite are listed.
detail	Optional. Specifies whether to list port and policy details for the Web service clients. Valid values are: ■ `true`—Output includes details about the clients, ports, policies, and whether the endpoint is secure or not. ■ `false`—Output lists only the clients. The default is false.

14.2.5.3 Examples

The following example lists information for all Web service clients in the domain.

```
wls:/wls-domain/serverConfig>listWebServiceClients()
```

The following example lists the Web service clients for the application `jwsclient_1#1.10` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>listWebServiceClients('soainfra/soa1/jwsclient_
1#1.10')
```

The following example lists the Web service clients for the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>listWebServiceClients(None,'default/HelloWorld[1.0]')
```

The following example lists details for all of the Web service clients in the domain.

```
wls:/wls-domain/serverConfig>listWebServiceClients(None,None,true)
```

14.2.6 listWebServiceClientPorts

Command Category: Web Service and Client Management

Use with WLST: Online

14.2.6.1 Description

List the Web service port names and the endpoint URLs for Web service clients in an application or SOA composite.

The output will display the name of the Web service client/reference port. For example:

```
AppModuleServiceSoapHttpPort
```

14.2.6.2 Syntax

```
listWebServiceClientPorts(application,moduleOrCompName,moduleType,serviceRefName)
```

Argument	Definition
application	Name and path of the application for which you want to list the Web services port information. For example, `/domain/server/application#version_number` To list the client port information for an application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to list the Web service client port information. To list the client port information for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	Service reference name of the application or SOA composite for which you want to list the Web service client port information. When the client is an asynchronous Web service callback client, the `serviceRefName` argument must be set to `callback`.

14.2.6.3 Examples

The following example lists the client ports for the `WssUsernameClient` Web module in the `/soainfra/soa1/jwsclient_1#1.1.0` application. Note that the `moduleType` is set to `wsconn`, and the `serviceRefName` is set to `WssUsernameClient`.

```
wls:/soainfra/serverConfig> listWebServiceClientPorts
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient')
```

The following example lists the client ports in the `default/HelloWorld[1.0]` SOA composite. Note that the `moduleType` is set to `soa`, and the `serviceRefName` is set to `client`.

```
wls:/soainfra/serverConfig> listWebServiceClientPorts(None,
'default/HelloWorld[1.0]','soa','client')
```

14.2.7 listWebServiceClientStubProperties

Command Category: Web Service and Client Management

Use with WLST: Online

14.2.7.1 Description

List Web service client port stub properties for an application or SOA composite.

14.2.7.2 Syntax

```
listWebServiceClientStubProperties(application, moduleOrCompName, moduleType,
serviceRefName, portInfoName)
```

Argument	Definition
application	Name and path of the application for which you want to list the Web services client port stub properties. For example, `/domain/server/application#version_number` To list the client port stub properties information for an application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to list the Web services client port stub properties. To list the client port stub properties information for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	Service reference name of the application or SOA composite for which you want to list the Web service client port stub properties.
portInfoName	The name of the client port for which you want to list the stub properties.

14.2.7.3 Example

The following example lists the client port stub properties for the `JRFWssUsernamePort` port of the `WssUsernameClient` Web module in the `/soainfra/soa1/jwsclient_1#1.1.0` application. Note that the `moduleType` is set to `wsconn`, and the `serviceRefName` is set to `WssUsernameClient`.

```
wls:/soainfra/serverConfig>listWebServiceClientStubProperties
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort')
```

14.2.8 setWebServiceClientStubProperty

Command Category: Web Service and Client Management

Use with WLST: Online

14.2.8.1 Description

Set, change, or delete a single stub property of a Web service client port for an application or SOA composite.

14.2.8.2 Syntax

```
setWebServiceClientStubProperty(application, moduleOrCompName, moduleType,
serviceRefName,portInfoName,propName,[propValue])
```

Argument	Definition
application	Name and path of the application for which you want to set the Web services client port stub property. For example, `/domain/server/application#version_number` To set a client port stub property for an application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to set the Web services client port stub property. To set a client port stub property for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	Service reference name of the application or SOA composite for which you want to set the Web service client port stub property.
portInfoName	The name of the client port for which you want to set the stub property.
propName	Stub property name that you want to set, change, or delete. For example, `'keystore.recipient.alias'`.
propValue	Optional. The stub property value, for example `'orakey'`. To remove the property, specify a blank `""` value.

14.2.8.3 Example

The following example sets the client port stub property `keystore.recipient.alias` to the value `oracle` for the client port `JRFWssUsernamePort`. The port is a client port of the `WssUsernameClient` Web module in the `/soainfra/soa1/jwsclient_1#1.1.0` application. Note that the `moduleType` is set to `wsconn`, and the `serviceRefName` is set to `WssUsernameClient`.

```
wls:/soainfra/serverConfig>setWebServiceClientStubProperty
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort','keystore.recipient.alias','oracle')
```

14.2.9 setWebServiceClientStubProperties

Command Category: Web Service and Client Management

Use with WLST: Online

14.2.9.1 Description

Configure the set of stub properties of a Web service client port for an application or SOA composite.

This command configures or resets all of the stub properties for the Oracle WSM client security policy attached to the client. Each property that you list in the command is set to the value you specify. If a property that was previously set is not explicitly specified in this command, it is reset to the default for the property. If no default exists, the property is removed.

14.2.9.2 Syntax

```
setWebServiceClientStubProperties(application, moduleOrCompName, moduleType,
serviceRefName, portInfoName, properties)
```

Argument	Definition
application	Name and path of the application for which you want to reset the Web services client port stub properties. For example, `/domain/server/application#version_number` To configure or reset the client port stub properties for an application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to reset the Web services client port stub properties. To configure or reset client port stub properties for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	Service reference name of the application or SOA composite for which you want to reset the Web service client port stub properties.
portInfoName	The name of the client port for which you want to reset the stub properties.

Argument	Definition

properties	The list of properties to be set or changed. Properties must be specified using the following format:
	`("property","value")`
	For example:
	`[("keystore.recipient.alias","oracle"), ("csf-key","oracle")]`
	To remove a property or clear the value assigned to it, specify a blank `""` value. For example:
	`[("csf-key","")]`
	To remove all the properties of the client port, set this argument to `None`.
	Sample client port stub properties are as follows:
	▪ `oracle.webservices.auth.username`
	▪ `oracle.webservices.auth.password`
	▪ `keystore.recipient.alias`
	▪ `csf-key`
	▪ `saml.issuer.name`
	▪ `javax.xml.ws.session.maintain`
	▪ `wsat.Version` —SOA references only
	▪ `wsat.flowOption`—SOA references only
	For a complete list of the configurable properties, see "Configuring the Web Service Client" in *Security and Administrator's Guide for Web Services*.

14.2.9.3 Example

The following example resets the client port stub properties `wsat.flowOption` and `wsat.Version` to `SUPPORTS` and `DEFAULT`, respectively. Any other properties that were previously set for this client port are either reset to the default or removed. The client port is `JRFWssUsernamePort` of the `WssUsernameClient` Web module in the `/soainfra/soa1/jwsclient_1#1.1.0` application. Note that the `moduleType` is set to `wsconn`, and the `serviceRefName` is set to `WssUsernameClient`.

```
wls:/soainfra/serverConfig>setWebServiceClientStubProperties('/soainfra/soa1/jwscl
ient_1#1.1.0',
'WssUsernameClient','wsconn','WssUsernameClient','JRFWssUsernamePort',
[("wsat.flowOption","SUPPORTS"),("wsat.Version","DEFAULT")] )
```

14.3 Policy Management Commands

Use the WLST commands listed in Table 14–3 to manage directly-attached Oracle WSM Web service and client policies.

When you set or change an attached policy in ADF and WebCenter Web service and client applications, you must restart the application for the changes to take effect. After the policy change is completed, a reminder message is displayed prompting you to restart the application. You can stop and restart the application using the standard `stopApplication` and `startApplication` WLST commands. For more information about these commands, see "Deployment Commands" on page 3-19.

Table 14–3 Web Services WLST Directly-attached Policy Management Commands

Use this command...	To...	Use with WLST...
listAvailableWebServicePolicies	Display a list of all the available Oracle Web Services Manager (WSM) policies by category or subject type.	Online
listWebServicePolicies	List Web service port policy information for a Web service in an application or SOA composite.	Online
attachWebServicePolicy	Attach a policy to a Web service port of an application or SOA composite.	Online
attachWebServicePolicies	Attach multiple policies to a Web service port of an application or SOA composite.	Online
enableWebServicePolicy	Enable or disable a policy attached to a port of a Web service application or SOA composite.	Online
enableWebServicePolicies	Enable or disable multiple policies attached to a port of a Web service application or SOA composite.	Online
detachWebServicePolicy	Detach an Oracle WSM policy from a Web service port of an application or SOA composite.	Online
detachWebServicePolicies	Detach multiple Oracle WSM policies from a Web service port of an application or SOA composite.	Online
listWebServiceClientPolicies	List Web service client port policies information for an application or SOA composite.	Online
attachWebServiceClientPolicy	Attach an Oracle WSM policy to a Web service client port of an application or SOA composite.	Online
attachWebServiceClientPolicies	Attach multiple policies to a Web service client port of an application or SOA composite.	Online
enableWebServiceClientPolicy	Enable or disable a policy of a Web service client port of an application or SOA composite.	Online
enableWebServiceClientPolicies	Enable or disable multiple policies of a Web service client port of an application or SOA composite.	Online
detachWebServiceClientPolicy	Detach a policy from a Web service client port of an application or SOA composite.	Online
detachWebServiceClientPolicies	Detach multiple policies from a Web service client port of an application or SOA composite.	Online
setWebServicePolicyOverride	Configure the Web service port policy override properties of an application or SOA composite.	Online

14.3.1 listAvailableWebServicePolicies

Command Category: Policy Management

Use with WLST: Online

14.3.1.1 Description

Display a list of all the available Oracle Web Services Manager (WSM) policies by category or subject type.

14.3.1.2 Syntax

```
listAvailableWebServicePolicies([category],[subject])
```

Argument	Definition
category	Optional. The policy category, for example: `'security','management'`.
subject	Optional. The policy subject type, for example: `'server'` or `'client'`.

14.3.1.3 Example

The following example lists all the available Oracle WSM server security policies in the domain.

```
wls:/wls-domain/serverConfig>listAvailableWebServicePolicies('security','server')
```

14.3.2 listWebServicePolicies

Command Category: Policy Management

Use with WLST: Online

14.3.2.1 Description

List Web service port policy information for a Web service in an application or SOA composite.

The output will display the Web service port name, the OWSM policies it has attached to it, and if applicable, any policy override properties. For example:

```
HelloWorldPort:
security : oracle/wss_username_token_service_policy , enabled=true
```

14.3.2.2 Syntax

```
listWebServicePolicies(application,moduleOrCompName,moduleType,serviceName,subject
Name)
```

Argument	Definition
application	Name and path of the application for which you want to list the Web services port policy information. For example, `/domain/server/application#version_number` To list the port policy information for a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to list the Web services port policy information. To list the port policy information for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.

Argument	Definition
moduleType	Module type. Valid options are: ■ web—Use with a Web service application (including EJB Web services). ■ soa—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite for which you want to list the port policy information. For example, {http://namespace/}serviceName
subjectName	Policy subject, port, or operation name.

14.3.2.3 Examples

The following example lists the Web service port policy information for the application HelloWorld#1_0 for the server soa1 in the domain soainfra. In this example, the Web module name is j2wbasicPolicy, the service name is WssUsernameService, and the subject is a port named JRFWssUsernamePort.

```
wls:/wls-domain/serverConfig>listWebServicePolicies
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort')
```

The following example lists the port policy information for the SOA composite default/HelloWorld[1.0]. Note that the moduleType is set to SOA, the service name is HelloService, and the subject is a port named HelloWorld_pt.

```
wls:/wls-domain/serverConfig>listWebServicePolicies
(None,
'default/HelloWorld[1.0]','soa','{http://namespace/}HelloService','HelloWorld_pt')
```

14.3.3 attachWebServicePolicy

Command Category: Policy Management

Use with WLST: Online

14.3.3.1 Description

Attach a policy to a Web service port of an application or SOA composite.

The policyURI is validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. If the PolicyURI that you specify in this command already is attached or exists, then this command enables the policy if it is disabled.

If the wsm-pm application is not installed or is not available, this command is not executed.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.3.2 Syntax

```
attachWebServicePolicy(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURI, [subjectType=None])
```

Argument	Definition
application	Name and path of the application to which you want to attach a Web service policy. For example, `/domain/server/application#version_number` To attach a policy to a port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) to which you want to attach a Web service policy. To attach a policy to a port of a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with a Web service application (including EJB Web services). ■ `soa`—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite. For example, `{http://namespace/}serviceName`
subjectName	Name of the policy subject, port, or operation.
policyURI	Oracle WSM policy name URI, for example `'oracle/log_policy'`
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.3.3 Examples

The following example attaches the policy `oracle/wss_username_token_service_policy` to the port `JRFWssUsernamePort` of the Web module `WssUsernameService`. The Web service is part of the application `HelloWorld#1_0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>attachWebServicePolicy
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort','oracle/wss_username_
token_service_policy')
```

The following example attaches the policy `oracle/log_policy` to the port `HelloWorld_pt` of the service `HelloService` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>attachWebServicePolicy(None,
'default/HelloWorld[1.0]',
'soa','{http://namespace/}HelloService','HelloWorld_pt','oracle/log_policy')
```

14.3.4 attachWebServicePolicies

Command Category: Policy Management

Use with WLST: Online

14.3.4.1 Description

Attach multiple policies to a Web service port of an application or SOA composite.

The policyURIs are validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. If any of the policies that you specify in this command are already attached or exist, then this command enables the policies that are already attached (if they are disabled), and attaches the others.

If the wsm-pm application is not installed or is not available, this command is not executed.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.4.2 Syntax

```
attachWebServicePolicies(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURIs, [subjectType=None])
```

Argument	Definition
application	Name and path of the application to which you want to attach the Web service policies. For example, `/domain/server/application#version_number` To attach the policies to a port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) to which you want to attach Web service policies. To attach the policies to a port of a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with a Web service application (including EJB Web services). ■ `soa`—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite. For example, `{http://namespace/}serviceName`
subjectName	Name of the policy subject, port, or operation.
policyURIs	List of Oracle WSM policy name URIs, for example `["oracle/log_policy","oracle/wss_username_token_service_policy"]` If any of the policies that you specify are already attached or exist, then this command enables the policies that are already attached (if they are disabled), and attaches the others.
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.4.3 Example

The following example attaches the policies `"oracle/log_policy"`, `"oracle/wss_username_token_service_policy"` to the port `JRFWssUsernamePort` of the Web module `WssUsernameService`. The Web service

is part of the application `HelloWorld#1_0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>attachWebServicePolicies
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort',
["oracle/log_policy", "oracle/wss_username_token_service_policy"])
```

14.3.5 enableWebServicePolicy

Command Category: Policy Management

Use with WLST: Online

14.3.5.1 Description

Enable or disable a policy attached to a port of a Web service application or SOA composite.

If the policy that you specify in this command is not attached to the port, an error message is displayed and/or an exception is thrown.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.5.2 Syntax

```
enableWebServicePolicy(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURI, [enable], [subjectType=None] ))
```

Argument	Definition
application	Name and path of the application for which you want to enable a Web service policy. For example, `/domain/server/application#version_number` To enable a policy that is attached to a port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to enable a Web service policy. To enable a policy that is attached to a port of a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with a Web service application (including EJB Web services). ■ `soa`—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite. For example, `{http://namespace/}serviceName`
subjectName	Name of the policy subject, port, or operation.

Argument	Definition
policyURI	Oracle WSM policy name URI, for example `'oracle/log_policy'` If the policy that you specify is not attached, an error message is displayed and/or an exception is thrown.
enable	Optional. Specifies whether to enable or disable the policy. Valid options are: ■ `true`—Enables the policy. The default is `true`. ■ `false`—Disables the policy. If you omit this argument, the policy is enabled.
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.5.3 Examples

The following example enables the policy `oracle/wss_username_token_service_policy` attached to the port `JRFWssUsernamePort` of the Web module `WssUsernameService`. The Web service is part of the application `HelloWorld#1_0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>enableWebServicePolicy
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort',"oracle/wss_username_
token_service_policy",true)
```

The following example enables the policy `oracle/log_policy` attached to the port `HelloWorld_pt` for the service `HelloService` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>enableWebServicePolicy(None,
'default/HelloWorld[1.0]',
'soa','{http://namespace/}HelloService','HelloWorld_pt','oracle/log_policy')
```

The following example disables the policy `oracle/log_policy` attached to the port `HelloWorld_pt` for the service `HelloService` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>enableWebServicePolicy(None,
'default/HelloWorld[1.0]',
'soa','{http://namespace/}HelloService','HelloWorld_pt','oracle/log_policy',false)
```

14.3.6 enableWebServicePolicies

Command Category: Policy Management

Use with WLST: Online

14.3.6.1 Description

Enable or disable multiple policies attached to a port of a Web service application or SOA composite.

If the policyURIs that you specify in this command are not attached to the port, an error message is displayed and/or an exception is thrown.

Note: Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.6.2 Syntax

```
enableWebServicePolicies(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURIs,[enable],[subjectType=None] ))
```

Argument	Definition
application	Name and path of the application for which you want to enable the Web service policies. For example, `/domain/server/application#version_number` To enable policies that are attached to a port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to enable Web service policies. To enable policies that are attached to a port of a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with a Web service application (including EJB Web services). ■ `soa`—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite.For example, `{http://namespace/}serviceName`
subjectName	Name of the policy subject, port, or operation.
policyURIs	List of Oracle WSM policy name URIs, for example `["oracle/log_policy","oracle/wss_username_token_service_policy"]` If the policyURIs that you specify are not attached, an error message is displayed and/or an exception is thrown.
enable	Optional. Specifies whether to enable or disable the policies. Valid options are: ■ `true`—Enables the policies. The default is `true`. ■ `false`—Disables the policies. If you omit this argument, the policies are enabled.
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.6.3 Example

The following example enables the policies `["oracle/log_policy","oracle/wss_username_token_service_policy"]` attached to the port `JRFWssUsernamePort` of the Web module `WssUsernameService`. The Web service is part of the application `HelloWorld#1_0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>enableWebServicePolicy
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort',["oracle/log_policy",
"oracle/wss_username_token_service_policy"],true)
```

14.3.7 detachWebServicePolicy

Command Category: Policy Management

Use with WLST: Online

14.3.7.1 Description

Detach an Oracle WSM policy from a Web service port of an application or SOA composite.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.7.2 Syntax

```
detachWebServicePolicy(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURI, [subjectType=None])
```

Argument	Definition
application	Name and path of the application from which you want to detach a Web service policy. For example, `/domain/server/application#version_number` To detach a policy from a port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) from which you want to detach a Web service policy. To detach a policy from a port of a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with a Web service application (including EJB Web services). ■ `soa`—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite. For example, `{http://namespace/}serviceName`
subjectName	Name of the policy subject, port, or operation.
policyURI	Oracle WSM policy name URI, for example `'oracle/log_policy'` If the policy specified is not attached, an error message is displayed and/or an exception is thrown.
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.7.3 Examples

The following example detaches the policy `oracle/wss_username_token_service_policy` from the port `JRFWssUsernamePort` of the Web module `WssUsernameService`. The Web service is part of the application `HelloWorld#1_0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>detachWebServicePolicy
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort','oracle/wss_username_
token_service_policy')
```

The following example detaches the policy `oracle/log_policy` from the port `HelloWorld_pt` of the service `HelloService` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>detachWebServicePolicy(None,
'default/HelloWorld[1.0]',
'soa','{http://namespace/}HelloService','HelloWorld_pt','oracle/log_policy')
```

14.3.8 detachWebServicePolicies

Command Category: Policy Management

Use with WLST: Online

14.3.8.1 Description

Detach multiple Oracle WSM policies from a Web service port of an application or SOA composite.

If the wsm-pm application is not installed or is not available, this command is not executed.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.8.2 Syntax

```
detachWebServicePolicies(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURIs, [subjectType=None])
```

Argument	Definition
application	Name and path of the application from which you want to detach the Web service policies. For example, `/domain/server/application#version_number` To detach policies from a port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) from which you want to detach the Web service policies. To detach policies from a port of a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.

Argument	Definition
moduleType	Module type. Valid options are: ■ web—Use with a Web service application (including EJB Web services). ■ soa—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite. For example, {http://namespace/}serviceName
subjectName	Name of the policy subject, port, or operation.
policyURIs	List of Oracle WSM policy name URIs, for example ["oracle/log_policy","oracle/wss_username_token_service_policy"] If a policyURI specified is not attached, an error message is displayed and/or an exception is thrown.
subjectType	Optional. Policy subject type. Valid options are: ■ P—Port. The default is P. ■ O—Not supported in this release.

14.3.8.3 Example

The following example detaches the policies "oracle/log_policy", "oracle/wss_username_token_service_policy" from the port JRFWssUsernamePort of the Web module WssUsernameService. The Web service is part of the application HelloWorld#1_0 for the server soa1 in the domain soainfra.

```
wls:/wls-domain/serverConfig>detachWebServicePolicies
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort',
["oracle/log_policy","oracle/wss_username_token_service_policy"])
```

14.3.9 listWebServiceClientPolicies

Command Category: Policy Management

Use with WLST: Online

14.3.9.1 Description

List Web service client port policies information for an application or SOA composite.

The output will display the Web service client/reference port name and the Oracle WSM policies it has attached to it. For example:

```
test-port:
security: oracle/wss_username_token_client_policy, enabled=true
```

14.3.9.2 Syntax

```
listWebServiceClientPolicies(application, moduleOrCompName, moduleType,
serviceRefName,portInfoName)
```

Argument	Definition
application	Name and path of the application for which you want to list the Web service client port policy information. For example, `/domain/server/application#version_number` To list the client port policy information for a Web services application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to list the Web services port policy information. To list the client port policy information for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	The service reference name of the application or composite.
portInfoName	The client port name.

14.3.9.3 Example

The following example lists the Web service client port policy information for the application `jwsclient_1#1.1.0` for the server `soa1` in the domain `soainfra`. In this example, the Web module name is `WssUsernameClient`, the module type is `wsconn`, the service reference name is `WssUsernameClient`, and the client port name is `JRFWssUsernamePort`.

```
wls:/wls-domain/serverConfig>listWebServiceClientPolicies
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort')
```

14.3.10 attachWebServiceClientPolicy

Command Category: Policy Management

Use with WLST: Online

14.3.10.1 Description

Attach a Oracle WSM policy to a Web service client port of an application or SOA composite.

The policyURI is validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. If the PolicyURI that you specify in this command already is attached or exists, then this command enables the policy if it is disabled.

If the wsm-pm application is not installed or is not available, this command is not executed.

Note: Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.10.2 Syntax

```
attachWebServiceClientPolicy(application,moduleOrCompName,moduleType,
serviceRefName, portInfoName, policyURI, [subjectType=None] )
```

Argument	Definition
application	Name and path of the application for which you want to attach a policy to the Web service client port. For example, `/domain/server/application#version_number` To attach a policy to a client port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to attach the policy to the client port. To attach a policy to a client port of a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	The service reference name of the application or composite.
portInfoName	The client port to which you want to attach the Oracle WSM client policy.
policyURI	The Oracle WSM policy name URI, for example `oracle/wss_username_token_client_policy"` If the policy that you specify is already attached or exists, then this command enables the policy if it is disabled.
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.10.3 Examples

The following example attaches the client policy `oracle/wss_username_token_client_policy` to the port `JRFWssUsernamePort` of the Web module `WssUsernameClient`. The Web service is part of the application `jwsclient_1#1.1.0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>attachWebServiceClientPolicy
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort',"oracle/wss_username_token_client_
policy")
```

The following example attaches the client policy `oracle/log_policy` to the client port `HelloWorld_pt` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>attachWebServiceClientPolicy
(None, 'default/HelloWorld[1.0]','soa','client','HelloWorld_pt','oracle/log_
policy')
```

14.3.11 attachWebServiceClientPolicies

Command Category: Policy Management

Use with WLST: Online

14.3.11.1 Description

Attach multiple policies to a Web service client port of an application or SOA composite.

The policyURIs are validated through the Oracle WSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available. If the policies that you specify in this command are already attached or exist, then this command enables the policies that are already attached (if they are disabled), and attaches the others.

If the wsm-pm application is not installed or is not available, this command is not executed.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.11.2 Syntax

```
attachWebServiceClientPolicies(application,moduleOrCompName,moduleType,
serviceRefName,portInfoName,policyURIs, [subjectType=None] )
```

Argument	Definition
application	Name and path of the application for which you want to attach Oracle WSM client policies to the Web service client port. For example, `/domain/server/application#version_number` To attach policies to a client port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to attach the policies to the client port. To attach policies to a client port of a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ▪ `web`—Use with asynchronous Web service callback client. ▪ `soa`—Required for a SOA composite. ▪ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	The service reference name of the application or composite.

Argument	Definition
portInfoName	The client port to which you want to attach the Oracle WSM client policy.
policyURI	The Oracle WSM policy name URIs, for example `["oracle/log_policy","oracle/wss_username_token_client_policy"]` If the policies that you specify in this command are already attached or exist, then this command enables the policies that are already attached (if they are disabled), and attaches the others.
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.11.3 Examples

The following example attaches the policy `oracle/wss_username_token_client_policy` to the port `JRFWssUsernamePort` of the Web module `WssUsernameClient`. The Web service is part of the application `jwsclient_1#1.1.0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>attachWebServiceClientPolicy
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort',"oracle/wss_username_token_client_
policy")
```

The following example attaches the policy `oracle/log_policy` to the client port `HelloWorld_pt` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>attachWebServiceClientPolicy
(None, 'default/HelloWorld[1.0]','soa','client','HelloWorld_pt','oracle/log_
policy')
```

14.3.12 enableWebServiceClientPolicy

Command Category: Policy Management

Use with WLST: Online

14.3.12.1 Description

Enable or disable a policy of a Web service client port of an application or SOA composite.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.12.2 Syntax

```
enableWebServiceClientPolicy(application,moduleOrCompName,moduleType,
serviceRefName,portInfoName,policyURI,[enable],[subjectType=None] )
```

Argument	Definition
application	Name and path of the application for which you want to enable or disable a policy of a Web service client port. For example, `/domain/server/application#version_number` To enable or disable a policy of a client port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to enable or disable a policy of a client port. To enable or disable a policy of a client port for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	The service reference name of the application or composite.
portInfoName	The name of the client port to which you want to attach the Oracle WSM client policy.
policyURI	The Oracle WSM policy name URI, for example `oracle/wss_username_token_client_policy"`
enable	Optional. Specifies whether to enable or disable the policy. Valid options are: ■ `true`—Enables the policy. The default is `true`. ■ `false`—Disables the policy. If you omit this argument, the policy is enabled.
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.12.3 Examples

The following example enables the client policy `oracle/wss_username_token_client_policy` of the port `JRFWssUsernamePort` of the Web module `WssUsernameClient`. The Web service is part of the application `jwsclient_1#1.1.0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>enableWebServiceClientPolicy
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort', "oracle/wss_username_token_client_
policy",true)
```

The following example enables the client policy `oracle/log_policy` of the client port `HelloWorld_pt` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>enableWebServiceClientPolicy(None,
'default/HelloWorld[1.0]','soa','client','HelloWorld_pt','oracle/log_policy')
```

The following example disables the client policy `oracle/log_policy` of the client port `HelloWorld_pt` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>enableWebServiceClientPolicy(None,
'default/HelloWorld[1.0]','soa','client','HelloWorld_pt','oracle/log_policy',
false )
```

14.3.13 enableWebServiceClientPolicies

Command Category: Policy Management

Use with WLST: Online

14.3.13.1 Description

Enable or disable multiple policies of a Web service client port of an application or SOA composite.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.13.2 Syntax

```
enableWebServiceClientPolicies(application,moduleOrCompName,moduleType,
serviceRefName,portInfoName,policyURIs,[enable], [subjectType=None] )
```

Argument	Definition
application	Name and path of the application for which you want to enable or disable multiple policies of a Web service client port. For example, `/domain/server/application#version_number` To enable or disable multiple policies of a client port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to enable or disable multiple policies of a client port. To enable or disable multiple policies of a client port for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	The service reference name of the application or composite.
portInfoName	The name of the client port to which you want to attach the Oracle WSM client policies.
policyURIs	The list of Oracle WSM policy name URIs, for example `["oracle/log_policy","oracle/wss_username_token_client_policy"]`.

Argument	Definition
enable	Optional. Specifies whether to enable or disable the policies. Valid options are: ■ true—Enables the policy. The default is true. ■ false—Disables the policy. If you omit this argument, the policies are enabled.
subjectType	Optional. Policy subject type. Valid options are: ■ P—Port. The default is P. ■ O—Not supported in this release.

14.3.13.3 Example

The following example enables the client policies oracle/log_policy and oracle/wss_username_token_client_policy of the port JRFWssUsernamePort of the Web module WssUsernameClient. The Web service is part of the application jwsclient_1#1.1.0 for the server soa1 in the domain soainfra.

```
wls:/wls-domain/serverConfig>enableWebServiceClientPolicies
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort',
["oracle/log_policy", "oracle/wss_username_token_client_policy"], true )
```

14.3.14 detachWebServiceClientPolicy

Command Category: Policy Management

Use with WLST: Online

14.3.14.1 Description

Detach a policy from a Web service client port of an application or SOA composite.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.14.2 Syntax

```
detachWebServiceClientPolicy(application,moduleOrCompName,moduleType,
serviceRefName, portInfoName, policyURI, [subjectType=None] )
```

Argument	Definition
application	Name and path of the application for which you want to detach a policy from a Web service client port. For example, /domain/server/application#version_number To detach a policy from a client port of a Web service application, this argument is required.

Argument	Definition
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to detach the policy from a client port. To detach a policy from a client port of a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	The service reference name of the application or composite.
portInfoName	The client port from which you want to detach the Oracle WSM client policy.
policyURI	The Oracle WSM policy name URI, for example `oracle/wss_username_token_client_policy"` If the policy specified is not attached, an error message is displayed and/or an exception is thrown.
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.14.3 Examples

The following example detaches the client policy `oracle/wss_username_token_client_policy` from the port `JRFWssUsernamePort` of the Web module `WssUsernameClient`. The Web service is part of the application `jwsclient_1#1.1.0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>detachWebServiceClientPolicy
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort','oracle/wss_username_token_client_
policy')
```

The following example detaches the client policy `oracle/log_policy` from the client port `HelloWorld_pt` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>detachWebServiceClientPolicy(None,
'default/HelloWorld[1.0]','soa','client','HelloWorld_pt','oracle/log_policy' )
```

14.3.15 detachWebServiceClientPolicies

Command Category: Policy Management

Use with WLST: Online

14.3.15.1 Description

Detach multiple policies from a Web service client port of an application or SOA composite.

Note: Policy changes made using this WLST command are only effective after you restart your application. For ADF and WebCenter applications, a message is displayed to remind you to restart your application.

14.3.15.2 Syntax

```
detachWebServiceClientPolicies(application,moduleOrCompName,moduleType,
serviceRefName,portInfoName,policyURIs, [subjectType=None] )
```

Argument	Definition
application	Name and path of the application for which you want to detach multiple policies from a Web service client port. For example, `/domain/server/application#version_number` To detach multiple policies from a client port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to detach multiple policies from a client port. To detach multiple policies from a client port for a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with asynchronous Web service callback client. ■ `soa`—Required for a SOA composite. ■ `wsconn`—Use with a connection-based Web service client such as an ADF DC Web service client, ADF JAX-WS Indirection Proxy, or WebCenter client.
serviceRefName	The service reference name of the application or composite.
portInfoName	The client port from which you want to detach the Oracle WSM client policy.
policyURI	The Oracle WSM policy name URI, for example `oracle/wss_username_token_client_policy"` If the policy specified is not attached, an error message is displayed and/or an exception is thrown.
subjectType	Optional. Policy subject type. Valid options are: ■ `P`—Port. The default is `P`. ■ `O`—Not supported in this release.

14.3.15.3 Example

The following example detaches the client policies `oracle/log_policy` and `oracle/wss_username_token_client_policy` from the port `JRFWssUsernamePort` of the Web module `WssUsernameClient`. The Web service is part of the application `jwsclient_1#1.1.0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>detachWebServiceClientPolicies
('/soainfra/soa1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort',
["oracle/log_policy","oracle/wss_username_token_client_policy"])
```

14.3.16 setWebServicePolicyOverride

Command Category: Policy Management

Use with WLST: Online

14.3.16.1 Description

Configure the Web service port policy override properties of an application or SOA composite.

14.3.16.2 Syntax

```
setWebServicePolicyOverride(application,moduleOrCompName,moduleType, serviceName,
portName,policyURI,properties)
```

Argument	Definition
application	Name and path of the application for which you want to override the Web service port policy. For example, `/domain/server/application#version_number` To override properties on a policy attached to a port of a Web service application, this argument is required.
moduleOrCompName	Name of the Web module or SOA composite (for example `HelloWorld[1.0]`) for which you want to override a Web service port policy. To override properties on a policy attached to a SOA composite, the composite name is required (for example `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`.
moduleType	Module type. Valid options are: ■ `web`—Use with a Web service application. ■ `soa`—Required for a SOA composite.
serviceName	Name of the Web service in the application or SOA composite. For example, `{http://namespace/}serviceName`
subjectName	Name of the policy subject, port, or operation.
policyURI	Oracle WSM policy name URI, for example `'oracle/log_policy'` to which the override properties will be applied. If the policy specified is not attached, an error message is displayed and/or an exception is thrown.
properties	Policy override properties. Properties must be specified using the following format: `[("name","value")]` For example: `[("myprop","myval")]` If this argument is set to `None`, then all policy overrides are removed.

14.3.16.3 Examples

The following example configures the override properties for the policy `oracle/wss10_message_protection_service_policy` for the port `JRFWssUsernamePort` of the Web module `WssUsernameService`. The Web service is part of the application `HelloWorld#1_0` for the server `soa1` in the domain `soainfra`.

```
wls:/wls-domain/serverConfig>setWebServicePolicyOverride
```

```
('/soainfra/soa1/HelloWorld#1_0','j2wbasicPolicy',
'web','{http://namespace/}WssUsernameService','JRFWssUsernamePort',
"oracle/wss10_message_protection_service_policy",
[("keystore.sig.csf.key","sigkey")])
```

14.4 Policy Set Management Commands

Policy sets enhance the security and manageability of an enterprise by providing a mechanism to globally attach one or more policies to a subject type. Using policy sets, an administrator can specify a default set of policies to be enforced even if none are directly attached. For detailed information about determining the type and scope of resources a policy set can be attached to, see "Creating and Managing Policy Sets" in the *Security and Administrator's Guide for Web Services*.

All policy set creation, modification, or deletion commands must be performed in the context of a repository session. A repository session can only act on a single document.

Note: The procedures in this chapter apply to Oracle Infrastructure Web Services only.

To view the help for the WLST commands described in this section, connect to a running instance of the server and enter `help('wsmManage')`.

Use the WLST commands listed in Table 14–4 to manage globally-available WSM Web service policy sets.

Table 14–4 Web Services WLST Globally-available Policy Set Management Commands

Use this command...	To...	Use with WLST...
beginRepositorySession	Begin a session to modify the Oracle WSM Repository.	Online
commitRepositorySession	Write the contents of the current session to the Oracle WSM repository.	Online
describeRepositorySession	Describe the contents of the current repository session.	Online
abortRepositorySession	Abort the current Oracle WSM Repository modification session, discarding any changes that were made to the repository during the session.	Online
createPolicySet	Create a new, empty policy set.	Online
listPolicySets	List the policy sets in the repository.	Online
clonePolicySet	Clone a new policy set from an existing policy set.	Online
displayPolicySet	Display the configuration of a specified policy set.	Online
modifyPolicySet	Specify an existing policy set for modification in the current session.	Online
setPolicySetPolicyOverride	Add a configuration override to a policy reference in the current policy set.	Online
setPolicySetConstraint	Specify a run-time constraint value for a policy set selected within a session.	Online

Table 14–4 (Cont.) Web Services WLST Globally-available Policy Set Management

Use this command...	To...	Use with WLST...
enablePolicySet	Enable or disable a policy set.	Online
enablePolicySetPolicy	Enable or disable a policy attachment for a policy set using the policy's URI.	Online
setPolicySetDescription	Specify a description for the policy set selected within a session.	Online
validatePolicySet	Validate an existing policy set in the repository or in a session.	Online
deletePolicySet	Delete a specified policy set.	Online
deleteAllPolicySets	Delete all or selected policy sets from within the Oracle WSM repository.	Online
attachPolicySet	Attach a policy set to the specified resource scope.	Online
attachPolicySetPolicy	Attach a policy to a policy set using the policy's URI.	Online
detachPolicySetPolicy	Detach a policy from a policy set using the policy's URI.	Online
migrateAttachments	Migrate direct policy attachments to global policy attachments if they are identical.	Online

14.4.1 beginRepositorySession

Command Category: Policy Set Management

Use with WLST: Online

14.4.1.1 Description

Begin a session to modify the Oracle WSM Repository. A repository session can only act on a single document. An error will be displayed if there is already a current session.

14.4.1.2 Syntax

```
beginRepositorySession()
```

14.4.1.3 Example

The following example begins an Oracle WSM Repository modification session.

```
wls:/wls-domain/serverConfig>beginRepositorySession()
```

14.4.2 commitRepositorySession

Command Category: Policy Set Management

Use with WLST: Online

14.4.2.1 Description

Write the contents of the current session to the Oracle WSM Repository. Messages are displayed that describe what was committed. An error will be displayed if there is no current session.

14.4.2.2 Syntax

```
commitRepositorySession()
```

14.4.2.3 Example

The following example commits the current repository modification session.

```
wls:/wls-domain/serverConfig>commitRepositorySession()
```

14.4.3 describeRepositorySession

Command Category: Policy Set Management

Use with WLST: Online

14.4.3.1 Description

Describe the contents of the current session. This will either indicate that the session is empty or list the name of the document that is being updated, along with the type of update (create, modify, or delete). An error will be displayed if there is no current session.

14.4.3.2 Syntax

```
describeRepositorySession()
```

14.4.3.3 Example

The following example describes the current repository modification session.

```
wls:/wls-domain/serverConfig>describeRepositorySession()
```

14.4.4 abortRepositorySession

Command Category: Policy Set Management

Use with WLST: Online

14.4.4.1 Description

Abort the current Oracle WSM Repository modification session, discarding any changes that were made to the repository during the session.

14.4.4.2 Syntax

```
abortRepositorySession()
```

14.4.4.3 Example

The following example aborts the current Oracle WSM Repository session.

```
wls:/wls-domain/serverConfig>abortRepositorySession()
```

14.4.5 createPolicySet

Command Category: Policy Set Management

Use with WLST: Online

14.4.5.1 Description

Create a new, empty policy set within a repository session. When creating a new policy set, you must specify the type of policy subject that the policy set will apply to, and a supported expression that defines a valid resource scope in a supported format.

Issuing this command outside of a repository session will result in an error.

14.4.5.2 Syntax

```
createPolicySet(name, type,attachTo,[description=None],[enable='true'])
```

Argument	Definition
name	Name of the new, empty policy set.
type	The type of policy subject to which the new policy set applies. The type of policy subject must be one of the following values: ■ `sca-component`—SOA Component ■ `sca-reference`—SOA Reference ■ `sca-service`—SOA Service ■ `ws-service`—Web Service Endpoint ■ `ws-client`—Web Service Client ■ `ws-connection`—Web Service Connection ■ `ws-callback`—Asynchronous Callback Client
attachTo	Expression that attaches the policy set to the specified resource scope. For details about specifying the resource scope expression, see "Resource Scope" in *Security and Administrator's Guide for Web Services*.
description	Optional. Description of the new policy set. If no description is specified, then the description for a new policy set will be "Global policy attachments for <*type*>", where <*type*> is the subject type.
enable	Optional. Specifies whether to enable or disable the new policy set. Valid options are: ■ `true`—Enables the new policy set. The default is `true`. ■ `false`—Disables the new policy set. If you omit this argument, the policy set is enabled.

14.4.5.3 Example

The first example creates a new policy set and specifies the resource scope to only `ws-service` types (Web Service Endpoint) in the `base_domain` domain. The second example creates a new policy set, but also narrows the resource scope to only `sca-service` types (SOA Service) in the `soa_server1` server in the domain.

```
wls:/wls-domain/serverConfig>createPolicySet('myPolicySet','ws-service','Domain("base_domain")')
wls:/wls-domain/serverConfig>createPolicySet('myPolicySet','sca-service','Server("soa_server1")','My policySet')
```

14.4.6 listPolicySets

Command Category: Policy Set Management

Use with WLST: Online

14.4.6.1 Description

Lists the policy sets in the repository. This command will also display a policy set that is being created, modified, or deleted within the current session. You can list all the policy sets or limit the display to include only those that apply to specific policy subject resource types.

14.4.6.2 Syntax

```
listPolicySets([type=None])
```

Argument	Definition
type=None	Optional. Specifies the type of policy sets to be displayed. The policy subject resource type must be one of the following values: ■ `sca-component`—SOA Component ■ `sca-reference`—SOA Reference ■ `sca-service`—SOA Service ■ `ws-service`—Web Service Endpoint ■ `ws-client`—Web Service Client ■ `ws-connection`—Web Service Connection ■ `ws-callback`—Asynchronous Callback Client If this argument is set to `None`, then all the policy sets stored in the repository will be listed.

14.4.6.3 Example

The first two examples list policy sets by either the `sca-reference` or `ws-client` resource types. Whereas, the third example lists all the policy sets stored in the repository.

```
wls:/wls-domain/serverConfig>listPolicySets('sca-reference')
wls:/wls-domain/serverConfig>listPolicySets('ws-client')
wls:/wls-domain/serverConfig>listPolicySets()
```

14.4.7 clonePolicySet

Command Category: Policy Set Management

Use with WLST: Online

14.4.7.1 Description

Within a repository session, clone a new policy set from an existing policy set. When cloning an existing policy set, all values and attachments in the source policy set are copied into the new policy set, although you can supply a different expression identifying the resource scope. The expression must define a valid resource scope in a supported format.

Issuing this command outside of a repository session will result in an error.

14.4.7.2 Syntax

```
clonePolicySet(name, source,[attachTo=None],[description=None],[enable='true'])
```

Argument	Definition
name	Name of the new policy set clone.

Argument	Definition
source	Name of the source policy set that will be cloned.
attachTo=None	Optional. Expression that attaches the policy set to the specified resource scope. For details about specifying the resource scope expression, see "Resource Scope" in *Security and Administrator's Guide for Web Services*. If this argument is set to `None`, then the expression used in the source policy set to identify the scope of resources is retained.
description=None	Optional. Description for the new policy set. If this argument is set to `None`, then the description used in the source policy set is retained.
enable='true'	Optional. Specifies whether to enable or disable the policy set. Valid options are: ■ `true`—Enables the policy set. The default is `true`. ■ `false`—Disables the policy set. If you omit this argument, the policy set is enabled.

14.4.7.3 Example

The first example creates a policy set by cloning the existing *myPolicySet* policy set to create a new *mynewPolicySet*. The second example also creates a policy set, but narrows the resource scope to policy subjects in the specified `soa_server1` server in the domain.

```
wls:/wls-domain/serverConfig>clonePolicySet('myNewPolicySet','myPolicySet')
wls:/wls-domain/serverConfig>clonePolicySet('myNewPolicySet','myPolicySet','Server
("soa_server1")')
```

14.4.8 displayPolicySet

Command Category: Policy Set Management

Use with WLST: Online

14.4.8.1 Description

Display the configuration of a specified policy set. If the policy set is being modified in the current session, then that version will be displayed; otherwise, the latest version in the repository will be displayed. An error will display if the policy set does not exist.

This command can be issued outside of a repository session.

14.4.8.2 Syntax

```
displayPolicySet([name])
```

Argument	Definition
name	Optional. Name of the policy set to be displayed. If a name is not specified, the configuration of the policy set, if any, in the current session is displayed or an error message is displayed.

14.4.8.3 Example

The following example displays the configuration of the *myPolicySet* policy set.

```
wls:/wls-domain/serverConfig>displayPolicySet('myPolicySet')
```

14.4.9 modifyPolicySet

Command Category: Policy Set Management

Use with WLST: Online

14.4.9.1 Description

Specify a policy set for modification in the current repository session. The latest version of the named policy set will be loaded into the current session. If the session already contains a different policy set, then an error will be displayed; if the session already contains the named policy set, then no action will be taken. Subsequent attempts to modify the named policy set will show the current version in the session.

Issuing this command outside of a repository session will result in an error.

14.4.9.2 Syntax

```
modifyPolicySet(name)
```

Argument	Definition
name	Name of the policy set to be modified in the current session.

14.4.9.3 Example

The following example opens the *myPolicySet* policy set for modification in the current session.

```
wls:/wls-domain/serverConfig>modifyPolicySet('myPolicySet')
```

14.4.10 setPolicySetPolicyOverride

Command Category: Policy Set Management

Use with WLST: Online

14.4.10.1 Description

Add a configuration override, described by a `name`, `value` pair, to an attached policy reference in the current policy set. The `value` argument is optional. If the `value` argument is omitted, the property specified by the `name` argument is removed from the policy reference in the policy set. If the property specified by the `name` argument already exists and a `value` argument is provided, the current value is overwritten by the new value specified with the `value` argument.

Issuing this command outside of a repository session containing a policy set that is being created or modified results in an error.

14.4.10.2 Syntax

```
setPolicySetPolicyOverride(uri,name,[value=None])
```

Argument	Definition
URI	String representing the Oracle WSM policy URI, for example `'oracle/wss10_saml_token_service_policy'` to which the override properties will be applied.
name	String representing the name of the override property. For example: `['reference.priority']`

Argument	Definition
value	Optional. String representing the value of the property. If this argument is not specified, the property specified by the `name` argument, if it exists, is removed.

14.4.10.3 Example

The following example specifies a configuration override for the `reference.priority` property for the `oracle/wss10_saml_token_service_policy` to a value of `1`.

```
setPolicySetPolicyOverride('oracle/wss10_saml_token_service_policy',
'reference.priority','1')
```

The following example removes the property `reference.priority` from the `oracle/wss10_saml_token_service_policy` in the policy set.

```
setPolicySetPolicyOverride('oracle/wss10_saml_token_service_policy',
'reference.priority')
```

14.4.11 setPolicySetConstraint

Command Category: Policy Set Management

Use with WLST: Online

14.4.11.1 Description

Specify a run-time constraint value for a policy set selected within a session. Issuing this command outside of a repository session containing a policy set that is being created or modified will result in an error.

For more information, see "Specifying Run-time Constraints in Policy Sets" in *Security and Administrator's Guide for Web Services*.

14.4.11.2 Syntax

```
setPolicySetConstraint(constraint)
```

Argument	Definition
constraint	Expression that specifies the run-time context to which the policy set applies. If not specified, the policy set applies to all run-time contexts.

14.4.11.3 Example

The following example specifies that the policy set apply only to requests from external clients.

```
setPolicySetConstraint('HTTPHeader("VIRTUAL_HOST_TYPE","external")')
```

The following example specifies that the policy set apply only to requests from non-external clients.

```
setPolicySetConstraint('!HTTPHeader("VIRTUAL_HOST_TYPE","external")')
```

14.4.12 enablePolicySet

Command Category: Policy Set Management

Use with WLST: Online

14.4.12.1 Description

Enable or disable the current policy set within a repository session. If not specified, this command enables the policy set.

Issuing this command outside of a repository session containing a policy set that is being created or modified will result in an error.

14.4.12.2 Syntax

```
enablePolicySet([enable=True])
```

Argument	Definition
enable	Optional. Specifies whether to enable or disable the policy set. Valid options are: ■ `true`—Enables the policy set. The default is `true`. ■ `false`—Disables the policy set. If you omit this argument, the policy set is enabled.

14.4.12.3 Example

The following example enables the current policy set.

```
wls:/wls-domain/serverConfig>enablePolicySet(true)
```

14.4.13 enablePolicySetPolicy

Command Category: Policy Set Management

Use with WLST: Online

14.4.13.1 Description

Within a repository session, enable or disable the policy attachment, which is identified by the provided URI in the current policy set. If not specified, this command enables the policy set. An error displays if the identified policy is not currently attached to the policy set.

Issuing this command outside of a repository session containing a policy set that is being created or modified will result in an error.

14.4.13.2 Syntax

```
enablePolicySetPolicy(uri,[enable=true])
```

Argument	Definition
uri	URI specifying the policy attachment within the policy set.
enable	Optional. Specifies whether to enable or disable the policy attachment specified by the URI in the policy set. Valid options are: ■ `true`—Enables the specified policy attachment in the policy set. The default is `true`. ■ `false`—Disables specified policy attachment in the policy set. If you omit this argument, the policy set attachment is enabled.

14.4.13.3 Example

The following example disables the specified logging policy attachment within the current policy set.

```
wls:/wls-domain/serverConfig>enablePolicySetPolicy('/oracle/log_policy',false)
```

14.4.14 setPolicySetDescription

Command Category: Policy Set Management

Use with WLST: Online

14.4.14.1 Description

Specify a description for a policy set selected within a session.

Issuing this command outside of a repository session containing a policy set that is being created or modified will result in an error.

14.4.14.2 Syntax

```
setPolicySetDescription(description)
```

Argument	Definition
description	Describes a policy set.

14.4.14.3 Example

The following example creates a description for a policy set.

```
wls:/wls-domain/serverConfig>setPolicySetDescription('PolicySetDescription')
```

14.4.15 validatePolicySet

Command Category: Policy Set Management

Use with WLST: Online

14.4.15.1 Description

Validates an existing policy set. If a policy set name is provided, the command will validate the specified policy set. If no policy set name is specified, the command will validate the policy set in the current repository session.

An error message displays if the policy set does not exist, or a name is not provided and the session is not active, or if the Oracle WSM Repository does not contain a suitable policy set.

14.4.15.2 Syntax

```
validatePolicySet([name=None])
```

Argument	Definition
name	Optional. Name of the policy set to validate. If a name is not provided then the command will validate the policy set being created or modified in the current session.

14.4.15.3 Example

The first example validates the policy set in the current session. The second example validates the specified *myPolicySet* policy set.

```
wls:/wls-domain/serverConfig>validatePolicySet()
wls:/wls-domain/serverConfig>validatePolicySet('myPolicySet')
```

14.4.16 deletePolicySet

Command Category: Policy Set Management

Use with WLST: Online

14.4.16.1 Description

Delete a specified policy set within a repository session. If the session already contains a different policy set, an error will display. If the session already contains the named policy set, then a creation will be undone or a modification will be converted into a deletion.

Issuing this command outside of a repository session will result in an error.

14.4.16.2 Syntax

```
deletePolicySet(name)
```

Argument	Definition
name	Name of the policy set to be deleted.

14.4.16.3 Example

The following example deletes a specified *myPolicySet* policy set.

```
wls:/wls-domain/serverConfig>deletePolicySet('myPolicySet')
```

14.4.17 deleteAllPolicySets

Command Category: Policy Set Management

Use with WLST: Online

14.4.17.1 Description

Delete all or selected policy sets from within the Oracle WSM repository. You can specify whether to force deletion of all the policy sets, or prompt to select individual policy sets for deletion. If deletion of any policy set fails then this operation throws an exception and no policy sets are deleted.

14.4.17.2 Syntax

```
deleteAllPolicySets([mode])
```

Argument	Definition
mode	Optional. The action to be taken for performing policy set deletion. Valid options are: ■ `force`—Automatically delete all policy sets without prompting. ■ `prompt`—Request user confirmation for each policy set deletion. Available options are `yes`, `no`, and `cancel`. If you select `cancel` for any property set deletion, the operation is canceled and no policy sets are deleted. If no mode is specified, this argument defaults to `prompt` mode.

14.4.17.3 Examples

The following example automatically deletes all policy sets from the respository without prompting.

```
wls:/jrfServer_domain/serverConfig> deleteAllPolicySets("force")

Starting Operation deleteAllPolicySets ...

All policy sets were deleted successfully from repository.

deleteAllPolicySets Operation Completed.
```

The following examples delete selected policy sets from the repository.

```
wls:/jrfServer_domain/serverConfig> deleteAllPolicySets()
```

or

```
wls:/jrfServer_domain/serverConfig> deleteAllPolicySets('prompt')

Starting Operation deleteAllPolicySets ...

Policy Set Name: create_policyset_6
Select "create_policyset_6" for deletion (yes/no/cancel)? no
Policy Set Name: create_policyset_8
Select "create_policyset_8" for deletion (yes/no/cancel)? yes
Policy Set Name: create_policyset_21
Select "create_policyset_21" for deletion (yes/no/cancel)? no
Policy Set Name: create_policyset_10
Select "create_policyset_10" for deletion (yes/no/cancel)? yes

All the selected policy sets were deleted successfully from repository.

deleteAllPolicySets Operation Completed.
```

14.4.18 attachPolicySet

Command Category: Policy Set Management

Use with WLST: Online

14.4.18.1 Description

Within a repository session, set an expression that attaches a policy set to the specified resource scope. The expression must define a valid resource scope in a supported format.

Issuing this command outside of a repository session containing a policy set that is being created or modified will result in an error.

14.4.18.2 Syntax

```
attachPolicySet(expression)
```

Argument	Definition
expression	Expression that attaches the policy set to the specified resource scope. For details about specifying the resource scope expression, see "Resource Scope" in *Security and Administrator's Guide for Web Services*.

14.4.18.3 Example

The following example attaches a policy set to the specified `base_domain` resource.

```
wls:/wls-domain/serverConfig>attachPolicySet('Domain("base_domain")')
```

This example attaches a policy set to the specified `base_domain` **and** `managed_server` resources.

```
wls:/wls-domain/serverConfig>attachPolicySet('Domain("base_domain") and
Server("managed_server")')
```

14.4.19 attachPolicySetPolicy

Command Category: Policy Set Management

Use with WLST: Online

14.4.19.1 Description

Within a repository session, attach a policy, identified by a specified URI, to the current policy set.

Issuing this command outside of a repository session containing a policy set that is being created or modified will result in an error.

14.4.19.2 Syntax

```
attachPolicySetPolicy(uri)
```

Argument	Definition
uri	URI specifying the policy to attach to the current policy set. For example, `'oracle/log_policy'`.

14.4.19.3 Example

The following example attaches the Oracle WSM logging policy to the current policy set.

```
wls:/wls-domain/serverConfig>attachPolicySetPolicy('oracle/log_policy')
```

14.4.20 detachPolicySetPolicy

Command Category: Policy Set Management

Use with WLST: Online

14.4.20.1 Description

Within a repository session, detach a policy, identified by a specified URI, from the current policy set.

Issuing this command outside of a repository session containing a policy set that is being created or modified will result in an error.

14.4.20.2 Syntax

```
detachPolicySetPolicy(uri)
```

Argument	Definition
uri	URI specifying the policy to detach to the current policy set. For example, `oracle/log_policy'`.

14.4.20.3 Example

The following example detaches the Oracle WSM logging policy from the current policy set.

```
wls:/wls-domain/serverConfig>detachPolicySetPolicy('oracle/log_policy')
```

14.4.21 migrateAttachments

Command Category: Policy Set Management

Use with WLST: Online

14.4.21.1 Description

Migrates direct (local) policy attachments that are identical to the external global policy attachments that would otherwise be attached to each policy subject in the current domain. You can specify whether to force the migration, prompt for confirmation before each migration, or simply list the migrations that would occur. A direct policy attachment is identical if its URI is the same as one provided by a global policy attachment, and if it does not have any scoped configuration overrides.

> **Note:** A direct attachment with an unscoped override will be migrated but an attachment with a scoped override will not. This is because after running the `migrateAttachments()` command, the enforcement of the policies on *all* subjects remains the same, even though some policies are globally attached.

Whether forced or prompted, the command lists each direct policy attachment that is migrated. This output will identify the policy subject that was modified, the URI of the identical policy reference, and the name of the global policy attachment document that duplicated the direct attachment.

14.4.21.2 Syntax

```
migrateAttachments([mode])
```

Argument	Definition
mode	The action to be taken for each policy attachment that can be migrated. Valid options are: ▪ `force`—Automatically migrate all identical policy attachments without prompting. ▪ `preview`—List all policy attachments that can be migrated, but does not perform any migration. ▪ `prompt`—Request user confirmation before migrating each policy attachment. If no mode is specified, this argument defaults to `prompt` mode.

14.4.21.3 Example

The following examples describe how to use the repository attachment migration modes.

```
wls:/wls-domain/serverConfig>migrateAttachments()
wls:/wls-domain/serverConfig>migrateAttachments('force')
wls:/wls-domain/serverConfig>migrateAttachments('preview')
wls:/wls-domain/serverConfig>migrateAttachments('prompt')
```

14.5 Oracle WSM Repository Management Commands

Use the commands listed in Table 14–5 to manage the WSM documents stored in the Oracle WSM Repository. For additional information about upgrading or migrating documents in an Oracle WSM Repository, see "Upgrading the Oracle WSM Policies in the Repository" in the *Security and Administrator's Guide for Web Services*.

Additional MDS WLST commands are described in Chapter 8, "Metadata Services (MDS) Custom WLST Commands."

Table 14–5 Policy Repository Management Commands

Use this command...	To...	Use with WLST...
upgradeWSMPolicyRepository	Upgrade the Oracle WSM predefined policies stored in the Oracle WSM Repository with any new predefined policies that are provided in the latest installation of the Oracle Fusion Middleware software.	Online
resetWSMPolicyRepository	Delete the existing policies stored in the Oracle WSM Repository and refresh it with the latest set of predefined policies that are provided in the new installation of the Oracle Fusion Middleware software.	Online
exportRepository	Export a set of documents from the repository into a supported ZIP archive. If the specified archive already exists, you can choose whether to overwrite the archive or merge the documents into the existing archive.	Online

Table 14–5 (Cont.) Policy Repository Management Commands

Use this command...	To...	Use with WLST...
importRepository	Import a set of documents from a supported ZIP archive into the repository. You can provide the location of a file that describes how to map a physical information from the source environment to the target environment.	Online

14.5.1 upgradeWSMPolicyRepository

Command Category: Policy Repository Management

Use with WLST: Online

14.5.1.1 Description

Upgrade the Oracle WSM predefined policies stored in the Oracle WSM Repository with any new predefined policies that are provided in the latest installation of the Oracle Fusion Middleware software. If the repository is empty, all of the predefined policies included in the installation are loaded into the repository.

This command does not remove any existing predefined and user-defined custom policies in the repository. If a predefined policy has been modified or discontinued in a subsequent release, one of the following occurs:

- For policies that have been discontinued, a message is displayed listing the discontinued policies. In this case, Oracle recommends that you no longer reference the policies and remove them using Oracle Enterprise Manager.
- For policies that have changed in the subsequent release, a message is displayed listing the changed policies. Oracle recommends that you import the latest version of the policies using Oracle Enterprise Manager.

14.5.1.2 Syntax

```
upgradeWSMPolicyRepository()
```

14.5.1.3 Example

The following example upgrades the existing installation with policies provided in the latest release:

```
wls:/wls-domain/serverConfig>upgradeWSMPolicyRepository()
```

14.5.2 resetWSMPolicyRepository

Command Category: Policy Repository Management

Use with WLST: Online

14.5.2.1 Description

Delete the existing policies stored in the Oracle WSM Repository and refresh it with the latest set of predefined policies that are provided in the new installation of the Oracle Fusion Middleware software. You can use the `clearStore` argument to specify whether to delete all policies, including custom user policies, from the Oracle WSM Repository before loading the new predefined policies.

14.5.2.2 Syntax

```
resetWSMPolicyRepository([clearStore='false'])
```

Argument	Definition
clearStore='false'	Policies to be deleted. Valid values are: ■ `true`—All policies in the repository, including custom user policies, are deleted. ■ `false`—Only the predefined policies supplied by Oracle are deleted. The default is `false`.

14.5.2.3 Example

The following example deletes all the policies in the repository, including user policies, and adds the predefined policies provided in the current product installation:

```
wls:/wls-domain/serverConfig>resetWSMPolicyRepository(true)
```

14.5.3 exportRepository

Command Category: Policy Repository Management

Use with WLST: Online

14.5.3.1 Description

Export a set of documents from the Oracle WSM Repository into a supported ZIP archive. If the specified archive already exists, the following options are presented:

```
The specified archive already exists. Update existing archive?
Enter "yes" to merge documents into existing archive, "no" to overwrite,
or "cancel" to cancel the operation.
```

You can also specify a list of the documents to be exported, or use a search expression to find specific documents in the repository.

14.5.3.2 Syntax

```
exportRepository(archive,[documents=None],[expandReferences='false'])
```

Argument	Definition
archive	Name of the archive file. If the specified archive already exists, you can choose whether to overwrite the archive or merge the documents into the existing archive. During override, the original archive is backed up and a message describes the location of the backup archive.
documents=None	Optional. The documents to be exported to the archive. If no documents are specified, then all assertion templates, intents, policies, and policy sets will be exported. You can specify a list of the documents to be exported, or use a search expression to find specific documents in the repository.
expandReferences='false'	Optional. Specifies whether the policy references should be expanded during export.

14.5.3.3 Example

The following examples describe repository export sessions. The first example exports all Oracle WSM documents to the `policies.zip` file.

```
wls:/wls-domain/serverConfig>exportRepository("/tmp/policies.zip")
```

This example exports only the `sca-component`, `sca-reference`, and `sca-service` policy sets to the `policies.jar` file, and also expands the all policy references output during the export process.

```
wls:/wls-domain/serverConfig>exportRepository("/tmp/policies.jar",
["/policysets/sca_component,/policysets/sca_reference,/policysets/sca_service]",
true)
```

This example exports policy sets using wildcards to the `some_global_with_noreference_2` file.

```
wls:/wls-domain/serverConfig>exportRepository('./export/some_global_with_
noreference_2',
['policysets:global/web_%','policysets:global/web_ref%', 'policysets:global/web_
call%'], false)
```

14.5.4 importRepository

Command Category: Policy Repository Management

Use with WLST: Online

14.5.4.1 Description

Import a set of documents from a supported ZIP archive into the Oracle WSM Repository. You can use the `map` argument to provide the location of a file that describes how to map physical information from the source environment to the target environment. For example, you can use the map file to ensure that the attachment expression in a policy set document is updated to match the target environment, such as `Domain("foo")=Domain("bar")`

14.5.4.2 Syntax

```
importRepository(archive,[map=None],[generateMapFile='false'])
```

Argument	Definition
archive	Name of the archive file.
map=None	Optional. Location of a sample map file that describes how to map physical information from the source environment to the target environment. You can generate a new map file by setting the `generateMapFile` argument to `true`. If you specify a map file without setting the `generateMapFile` argument to `true`, and the file does not exist, the operation fails and an error is displayed.
generateMapFile=false	Optional. Specify whether to create a sample map file at the location specified by the map argument. No documents are imported when this argument is set to `true`. The default is `false`. After the file is created you can edit it using any text editor. The `attachTo` values can be updated according to the new environment details. If there is no update required for any document name, that entry may be either deleted or commented using the character ("#").

14.5.4.3 Example

The following examples describe repository import sessions.

The first example imports the contents of the `policies.zip` file into the repository.

```
wls:/wls-domain/serverConfig>importRepository("/tmp/policies.zip")
```

This example uses the `generateMapFile` argument to generate a map file.

```
wls:/wls-domain/serverConfig>importRepository("./export/some_global_with_
noreference_2', map="./export/some_global_with_noreference_2_map',
generateMapFile=true)
```

Here is an example of a generated map file:

```
This is an auto generated override file containing the document names given in
the archive file and their corresponding attachTo values. The attachTo value can
be updated according to the new environment details. If there is no update
required for any document name,that entry may be either deleted or commented
using the character ("#")

[Resource Scope Mappings
]
sca_component_add_1=Composite("*Async*")
sca_reference_add_1=Composite("*Basic_SOA_Client*")
sca_reference_no=Server("*")
sca_service_add_1=Composite("*Basic_SOA_service")
web_callback_add_1=Application("*")
web_client_add_1=Module("*")
web_reference_add_1=Domain("*")
web_service_add_1=Domain("*domain*") and Server("*soa*") and Application("*ADF*")
ws_service_no_1=Server("*Admin*")
```

This example illustrates how to import documents using a generated map file: `/some_global_with_noreference_2_map`.

```
wls:/wls-domain/serverConfig>importRepository('../export/export_all', 'export_all_
map')
```

14.6 Deployment Descriptor Migration Commands

Use the commands listed in Table 14–6 to migrate the ADF Business Components and WebCenter services proprietary deployment descriptor (PDD) files between environments, such as from test to production.

For additional information about using these commands, see "Managing Application Migration Between Environments" in the *Security and Administrator's Guide for Web Services*.

Table 14–6 Deployment Descriptor Migration Commands

Use this command...	To...	Use with WLST ...
exportJRFWSApplicationPDD	Export an ADF Business Control or WebCenter application deployment descriptor to a Java Archive (JAR) file.	Online
importJRFWSApplicationPDD	Import an ADF Business Control or WebCenter Web service application deployment descriptor from the exported JAR file into a new environment, for example, a production environment or a scaled server instance in a cluster.	Online

Table 14–6 (Cont.) Deployment Descriptor Migration Commands

Use this command...	To...	Use with WLST ...
savePddToAllAppInstancesInDomain	Import and save the ADF BC or WebCenter Web service application deployment descriptor from the exported JAR file into all of the server instances in the connected domain.	Online

14.6.1 exportJRFWSApplicationPDD

Command Category: Deployment descriptor migration

Use with WLST: Online

14.6.1.1 Description

Export an ADF Business Control or WebCenter application deployment descriptor to a Java Archive (JAR) file. If you do not specify a name for the JAR file, the output displays the default name and path to the JAR file.

14.6.1.2 Syntax

```
exportJRFWSApplicationPDD(application,pddJarFileName=None)
```

Argument	Definition
application	Name and path of the application for which you want to export the configuration information. For example, `/domain/server/application#version`
pddJarFileName	Optional. User-specified name for the JAR file. The default is `None`. For example, `/tmp/myPDD.jar`.

14.6.1.3 Example

The following example exports the Web service PDD for the application `ADFBCHelloWorld` into a JAR file named `exportPDD.jar`.

```
wls:/wls-domain/serverConfig>exportJRFWSApplicationPDD
('/wls-domain/ManagedServer/ADFBCHelloWorld','/tmp/exportPDD.jar')

/tmp/exportPDD.jar
```

14.6.2 importJRFWSApplicationPDD

Command Category: Deployment descriptor migration

Use with WLST: Online

14.6.2.1 Description

Import an ADF Business Control or WebCenter Web service application deployment descriptor from the exported JAR file into a new environment, for example, a production environment or a scaled server instance in a cluster.

Note: Changes made using this WLST command are only effective after you restart your application. After importing the deployment descriptor, a message is displayed to remind you to restart your application.

14.6.2.2 Syntax

```
importJRFWSApplicationPDD(application,pddJarFileName)
```

Argument	Definition
application	Fully qualified path and name of the application to which you want to import the configuration information. For example, `/domain/server/application#version`
pddJarFileName	Name of the JAR file that contains the PDD file to be imported. For example, `/tmp/myPDD.jar`

14.6.2.3 Example

The following example imports the Web service application deployment descriptor for the `ADFBCHelloWorld` application that has been migrated to the server `ManagedServer2`. The command uses the name of the JAR file that was generated when the `exportJRFWSApplicationPDD` command was executed.

```
wls:/wls-domain/serverConfig>importJRFWSApplicationPDD
('/wls-domain/ManagedServer2/ADFBCHelloWorld', '/tmp/exportPDD.jar')

application  /wls-domain/ManagedServer2/ADFBCHelloWorld PDD has been reset,
 please restart application now to uptake changes!
```

14.6.3 savePddToAllAppInstancesInDomain

Command Category: Deployment descriptor migration

Use with WLST: Online

14.6.3.1 Description

Import and save the ADF BC or WebCenter Web service application deployment descriptor from the exported JAR file into all of the server instances in the connected domain. You can also use the optional `restartApp` argument to restart the application automatically.

14.6.3.2 Syntax

```
savePddToAllAppInstancesInDomain(applicationName,pddJarFileName,restartApp=true)
```

Argument	Definition
applicationName	Name of the application to which you want to import the configuration information. For example, `application#version`
pddJarFileName	Name of the JAR file that contains the PDD file to be imported. For example, `/tmp/myPDD.jar`
restartApp	Optional. Restart the application. Valid values are: ■ `true`—Restart the application automatically. The default is `true`. ■ `false`—Do not restart the application automatically.

14.6.3.3 Example

The following example imports the Web service application deployment descriptor for the `ADFBCHelloWorld` application that was previously exported into all of the servers in the domain, and restarts the application.

```
wls:/wls-domain/serverConfig>savePddToAllAppInstancesInDomain
'ADFBCHelloWorld', '/tmp/exportPDD.jar' , true

saving pdd to  com.bea:ServerRuntime=ManagedServer,Name=ADFBCHelloWorld,
Location=ManagedServer,Type=ApplicationRuntime
saving pdd to  com.bea:ServerRuntime=ManagedServer2,Name=ADFBCHelloWorld,
Location=ManagedServer2,Type=ApplicationRuntime
restarting application  ADFBCHelloWorld
Stopping application ADFBCHelloWorld.
<Mar 24, 2010 10:50:07 AM PDT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating stop operation for application, ADFBCHelloWorld
[archive: null], to Cluster-1 .>
.Completed the stop of Application with status completed
Current Status of your Deployment:
Deployment command type: stop
Deployment State       : completed
Deployment Message     : no message
Starting application ADFBCHelloWorld.
<Mar 24, 2010 10:50:11 AM PDT> <Info> <J2EE Deployment SPI> <BEA-260121>
<Initiating start operation for application, ADFBCHelloWorld
[archive: null], to Cluster-1 .>
.Completed the start of Application with status completed
Current Status of your Deployment:
Deployment command type: start
Deployment State       : completed
Deployment Message     : no message
```

The following example imports the Web service application deployment descriptor for the `ADFBCHelloWorld` application that was previously exported into all of the servers in the domain, but does not restart the application automatically. This example shows the commands you need to enter to restart the application manually.

```
wls:/wls-domain/serverConfig>savePddToAllAppInstancesInDomain
('ADFBCHelloWorld', '/tmp/exportPDD.jar', false)

saving pdd to  com.bea:ServerRuntime=ManagedServer,Name=ADFBCHelloWorld,
Location=ManagedServer,Type=ApplicationRuntime
saving pdd to  com.bea:ServerRuntime=ManagedServer2,Name=ADFBCHelloWorld,
Location=ManagedServer2,Type=ApplicationRuntime
application  ADFBCHelloWorld  PDD has been reset, please restart application now
to uptake changes!

wls:/wls-domain/serverConfig> stopApplication('ADFBCHelloWorld')
wls:/wls-domain/serverConfig> startApplication('ADFBCHelloWorld')
```

15

Diagnostic Framework Custom WLST Commands

The Diagnostic Framework aids in capturing relevant and timely diagnostics for critical errors. The diagnostics can be sent to Oracle Support for further analysis. Use the Diagnostic Framework commands to generate incidents, query existing incidents and execute individual diagnostics dumps to gather specific diagnostics data.

For additional information about using the Diagnostic Framework, see "Diagnosing Problems" in the *Oracle Fusion Middleware Administrator's Guide.*

> **Note:** To use the Diagnostic Framework custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.*

Table 15–1 lists the different categories of Diagnostic Framework commands.

Table 15–1 Diagnostic Command Categories

Command Category	Description
Incident Commands	View problems and incidents and to create incidents.
Diagnostic Dump Commands	Display information about dumps and to execute dumps.

15.1 Incident Commands

Use the commands in Table 15–2 to view problems and incidents and to create incidents.

Table 15–2 Incident Commands

Use this command...	To...	Use with WLST...
createIncident	Create a diagnostic incident.	Online
getIncidentFile	Retrieves the contents of the specified incident file.	Online
listADRHomes	List the set of ADR Home paths.	Online
listIncidents	List a set of diagnostic incidents.	Online
listProblems	List a set of diagnostic problems.	Online
showIncident	Show the details of a specified incident.	Online

15.1.1 createIncident

Use with WLST: Online

15.1.1.1 Description

Creates a diagnostic incident, using the specified information to determine the set of diagnostic rules and actions to execute.

15.1.1.2 Syntax

```
createIncident([adrHome] [,incidentTime] [,messageId] [,ecid] [,appName]
  [,description] [,server])
```

Argument	Definition
adrHome	The path for the ADR Home in which to create the incident. The ADR Home must exist. If this argument is not specified, the default ADR Home is used. The default ADR Home is the following location: *ADR_BASE*/diag/OFM/*domain_name*/*server_name*
incidentTime	The timestamp at which the incident occurred. If this not specified, the current time is used. You can specify the following: ■ The time of the current day, in the format HH:MM. For example: 19:45 ■ The date and time, in the format MM/DD/YYYY HH:MM
messageId	The ID of the error message. For example, MDS-50400.
ecid	The Execution Context ID for the error message.
appNname	The name of the deployed application for which the diagnostics are being gathered. For example, if you have multiple ADF applications deployed, each may register a dump called adf.dump. To execute this command for a specific application, you must specify the application name.
description	Descriptive text to associate with the incident. This is useful when reviewing the incident at a later time.
server	The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.

15.1.1.3 Example

The following example creates an incident that is related to messages with the ID MDS-50400:

```
createIncident(messageId="MDS-50400", description="sample incident")
Incident Id: 55
Problem Id: 4
Problem Key: MDS-50400 [MANUAL]
Incident Time: 25th March 2010 11:55:45 GMT
Error Message Id: MDS-50400
Flood Controlled: false
```

15.1.2 getIncidentFile

Use with WLST: Online

15.1.2.1 Description

Retrieves the contents of the specified incident file.

15.1.2.2 Syntax

```
getIncidentFile(id, name [,outputFile] [,adrHome] [,server])
```

Argument	Definition
id	The ID of the incident that you want to retrieve.
name	The name of the file to retrieve. To find the name of the file, use the showIncident command.
outputFile	The name of the file to which to write the output.
adrHome	The path for the ADR Home from which to retrieve the information. If this argument is not specified, the default ADR Home will be queried. The default ADR Home is the following location: *ADR_BASE*/diag/OFM/*domain_name*/*server_name*
server	The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.

15.1.2.3 Example

The following example writes the contents of the incident dms_metrics3_i1.dmp to the specified output file:

```
getIncidentFile(id='1', name='dms_metrics3_i1.dmp', outputFile='/tmp/incident1_
dms.txt')
The content of 'dms_metrics3_i1.dmp'is written to /tmp/incident1_dms.txt
```

15.1.3 listADRHomes

Use with WLST: Online

15.1.3.1 Description

Lists the paths of all of the ADR Homes for the server.

15.1.3.2 Syntax

```
listADRHomes([server])
```

Argument	Definition
server	The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.

15.1.3.3 Example

The following example lists the paths of the ADR homes:

```
listADRHomes()
diag/ofm/base_domain/WLS_Spaces

diag/ofm/fusionapps/GeneralLedger
```

15.1.4 listIncidents

Use with WLST: Online

15.1.4.1 Description

Lists the set of diagnostic incidents for the given problem ID, if specified, or all available incidents.

15.1.4.2 Syntax

```
listIncidents([id] [, adrHome] [,server])
```

Argument	Definition
id	The ID of the problem for which you want to list the set of diagnostic incidents.
adrHome	The path for the ADR Home from which to query incidents. If this argument is not specified, the default ADR Home will be queried. The default ADR Home is the following location: *ADR_BASE*/diag/OFM/*domain_name*/*server_name*
server	The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.

15.1.4.3 Example

The following example lists the incidents associated with the problem with the ID `1`:

```
listIncidents(id="1")
Incident Id   Problem Key                                 Incident Time
         10   MDS-50300 [WLS_Spaces] [oracle.mds.repos]   Mon Mar 15 11:22:12 PDT
2010
         24   MDS-50300 [WLS_Spaces] [oracle.mds.repos]   Thu Mar 11 15:11:35 PDT
2010
```

15.1.5 listProblems

Use with WLST: Online

15.1.5.1 Description

Lists the set of diagnostic problems associated with the specified ADR Home.

15.1.5.2 Syntax

```
listProblems([adrHome][,server])
```

Argument	Definition
adrHome	The path for the ADR Home from which to query problems. If this argument is not specified, the default ADR Home will be queried. The default ADR Home is the following location: *ADR_BASE*/diag/OFM/*domain_name*/*server_name*
server	The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.

15.1.5.3 Example

The following example lists the diagnostic problems in the default ADR home:

```
listProblems()
Problem Id        Problem Key
         1        MDS-50300 [WLS_Spaces] [oracle.mds.repos]
         2        JOC-38922 [AdminServer] [oracle.cache.network]
```

15.1.6 showIncident

Use with WLST: Online

15.1.6.1 Description

Shows the details of the specified incident.

15.1.6.2 Syntax

```
showIncident(id, [adrHome][, server])
```

Argument	Definition
id	The ID of the incident that you want to view.
adrHome	The path for the ADR Home from which to query the incident. If this argument is not specified, the default ADR Home will be queried. The default ADR Home is the following location: *ADR_BASE*/diag/OFM/*domain_name*/*server_name*
server	The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.

15.1.6.3 Example

The following example displays information about the incident with the ID 10:

```
showIncident(id="10")
Incident Id: 10
Problem Id: 1
Problem Key: MDS-50300 [WLS_Spaces] [oracle.mds.repos]
Incident Time: 25th March 2010 10:12:15 GMT
Error Message Id: MDS-50300
Execution Context: 0000ICK4rbYC8xT6uBf9EH1AX1qF000000
Flood Controlled: false
Dump Files :
      dms_ecidctx1_i1.dmp
      jvm_threads2_i1.dmp
      dms_metrics3_i1.dmp
      odl_logs4_i1.dmp
      diagnostic_image_AdminServer_2010_03_25_11_12_15.zip
      readme.txt
```

15.2 Diagnostic Dump Commands

Use the commands in Table 15–3 to display information about dumps and to execute dumps.

Table 15–3 Diagnostic Dump Commands

Use this command...	To...	Use with WLST...
describeDump	Display a description of the specified diagnostic dump.	Online
executeDump	Execute the specified diagnostic dump.	Online
listDumps	Display the set of diagnostic dumps that can be executed.	Online

15.2.1 describeDump

Use with WLST: Online

15.2.1.1 Description

Displays a description of the specified diagnostic dump.

15.2.1.2 Syntax

```
describeDump(name [,appName] [.server])
```

Argument	Definition
name	The name of the dump for which to display information.
appName	The name of the deployed application for which information is gathered. For example, if you have multiple ADF applications deployed, each may register a dump called adf.dump. To execute this command for a specific application, you must specify the application name.
server	The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.

15.2.1.3 Example

The following example displays information about the dump with the name odl.logs. You use the listDumps command to retrieve the list of available dumps.

```
describeDump(name="odl.logs")
Name: odl.logs
Description: Dumps recent ODL logs, or logs correlated by ECID
Manadatory Arguments:
Optional Arguments:
   Name       Type      Description
   ECID       String    Execution Context Id to correlate log entries with
   timestamp  String    Timestamp to query logs 5 minutes before/after
```

15.2.2 executeDump

Use with WLST: Online

15.2.2.1 Description

Executes the specified diagnostic dump.

15.2.2.2 Syntax

```
executeDump(name [,args] [,outputFile] [,id] [,adrHome] [,server])
```

Argument	Definition
name	The name of the diagnostic dump to execute.
args	Mandatory or optional arguments to pass to the dump.
outputFile	The name of the file to which to write the dump. If you do not specify this argument, the output is written to the console.
id	The ID of the incident to which to associate the dump. By default, the specified dump will not be associated with an incident.
adrHome	The ADR home that contains the incident. If you do not specify this argument, the default ADR home is used. The default ADR Home is the following location: *ADR_BASE*/diag/OFM/*domain_name*/*server_name*
server	The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.

Arguments that are either required or are optional can be specified using the "args" keyword. For example:

```
executeDump("java.sysprops",args={"prop" : "os.name"})
```

15.2.2.3 Examples

The following example executes the dump with the name jvm.threads and writes it to the file dumpout.txt:

```
executeDump(name="jvm.threads", outputFile="/tmp/dumpout.txt")
Diagnostic dump jvm.threads output written to /tmp/dumpoutput.txt
```

The following example executes the dump with the name jvm.threads and the Incident ID for 33 and writes it to the file dumpout.txt:

```
executeDump(name="jvm.threads", outputFile="/tmp/dumpout.txt", id="33")
Diagnostic dump jvm.threads output associated with incident 33 in ADR Home
diag/ofm/base_domain/AdminServer
```

The following example executes a dump with the argument `prop` set to the value `os.name`:

```
executeDump("java.sysprops",args={"prop" : "os.name"})
```

15.2.3 listDumps

Use with WLST: Online

15.2.3.1 Description

Displays the set of diagnostic dumps that can be executed.

15.2.3.2 Syntax

```
listDumps([appName] [,server])
```

Argument	Definition
appName	The name of a deployed application for which diagnostics are being gathered. For example, if you have multiple ADF applications deployed, each may register a dump called adf.dump. To execute this command for a specific application, you must specify the application name. If you specify this argument, the command returns the dumps for the specified application. If you do not specify this argument, the command returns the system dumps.
server	The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server.

15.2.3.3 Example

The following example lists all of the available dumps.

```
listDumps()
dms.metrics
jvm.classhistogram
jvm.threads
odl.logs

Use the command describeDump(name=<dumpName>) for help on a specific dump.
```

16

Information Rights Management Custom WLST Commands

The following sections describe the Oracle Fusion Middleware Information Rights Management custom WLST commands in detail. Topics include:

- Section 16.1, "Overview of WLST IRM Commands"
- Section 16.2, "General Server Commands"
- Section 16.3, "Migration Commands"
- Section 16.4, "Test Content Commands"
- Section 16.5, "Languages Support Commands"
- Section 16.6, "Oracle IRM Desktop Installers Commands"

For additional information about Oracle Information Rights Management, see *Oracle IRM 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.*

16.1 Overview of WLST IRM Commands

WLST IRM commands are divided into the following categories:

Table 16–1 WLST IRM Command Categories

Command Category	Description
General Server Commands	Make general changes to Oracle IRM Server settings.
Migration Commands	Back up and migrate Oracle IRM Server user data.
Test Content Commands	Set up test content for users of Oracle IRM Desktop.
Languages Support Commands	Set up languages support for users of Oracle IRM Server.
Oracle IRM Desktop Installers Commands	Set up software installation support for Oracle IRM Desktop.

16.2 General Server Commands

Use the WLST commands listed in Table 16–2 to make general changes to Oracle IRM Server settings.

Table 16–2 WLST General Server Commands

Use this command...	To...	Use with WLST...
addIRMRefreshPeriod	Create a new refresh period.	Online
getIRMRefreshPeriod	Display an existing refresh period.	Online
getIRMRefreshPeriods	Display all the refresh periods.	Online
removeIRMRefreshPeriod	Remove an existing refresh period.	Online
updateIRMRefreshPeriod	Update an existing refresh period.	Online
addIRMSyncWindow	Create a new sync window.	Online
getIRMSyncWindow	Display an existing sync window.	Online
getIRMSyncWindows	Display all the sync windows.	Online
removeIRMSyncWindow	Remove an existing sync window.	Online
updateIRMSyncWindow	Update an existing sync window.	Online
getIRMCryptoSchema	Display the cryptography algorithm.	Online
setIRMCryptoSchema	Set the cryptography algorithm.	Online
getIRMDeviceCount	Display the device count.	Online
setIRMDeviceCount	Set the device count.	Online
getIRMJournalCleanUp	Display the current report record clean-up values.	Online
setIRMJournalCleanUp	Set report record clean-up values.	Online
getIRMLicenseStateCleanUp	Display the license state clean-up frequency.	Online
setIRMLicenseStateCleanUp	Set the license state clean-up frequency.	Online
getIRMPrivacyURL	Display the URL of the privacy statement page.	Online
setIRMPrivacyURL	Set the URL of the privacy statement page.	Online
getIRMKeyStore	Display the type and location of the Oracle IRM keystore.	Online
setIRMKeyStore	Set the type and location of the Oracle IRM keystore.	Online

16.2.1 addIRMRefreshPeriod

Online command that creates a new refresh period.

16.2.1.1 Description

This command creates a refresh period. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.1.2 Syntax

```
addIRMRefreshPeriod(duration,dtype)
```

Argument	Definition
duration	Specifies the value of the refresh period. Integer.
dtype	Specifies the unit of the refresh period. 'MINUTES', 'HOURS', 'DAYS', 'MONTHS', 'YEARS'.

16.2.1.3 Examples

The following example creates a refresh period of 5 hours:

```
wls:/base_domain/serverConfig> addIRMRefreshPeriod(5,\"HOURS\")
```

The following example creates a refresh period of 50 minutes:

```
wls:/base_domain/serverConfig> addIRMRefreshPeriod(50,\"MINUTES\")
```

16.2.2 getIRMRefreshPeriod

Online command that displays an existing refresh period.

16.2.2.1 Description

This command displays the refresh period that is present at the specified index. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.2.2 Syntax

```
getIRMRefreshPeriod(pindex)
```

Argument	Definition
pindex	Specifies the index of the refresh period.

16.2.2.3 Examples

The following example displays the refresh period that is present at index zero:

```
wls:/base_domain/serverConfig> getIRMRefreshPeriod(0)
```

The following example displays the refresh period that is present at index one:

```
wls:/base_domain/serverConfig> getIRMRefreshPeriod(1)
```

16.2.3 getIRMRefreshPeriods

Online command that displays all existing refresh periods.

16.2.3.1 Description

This command displays all existing refresh periods. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.3.2 Syntax

```
getIRMRefreshPeriods()
```

16.2.3.3 Example

```
wls:/base_domain/serverConfig> getIRMRefreshPeriods()
```

16.2.4 removeIRMRefreshPeriod

Online command that removes an existing refresh period.

16.2.4.1 Description

This command removes a refresh period that is present at the specified index. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.4.2 Syntax

```
removeIRMRefreshPeriod(pindex)
```

Argument	Definition
pindex	Specifies the index of the refresh period.

16.2.4.3 Examples

The following example removes the refresh period that is present at index zero:

```
wls:/base_domain/serverConfig> removeIRMRefreshPeriod(0)
```

The following example removes the refresh period that is present at index one:

```
wls:/base_domain/serverConfig> removeIRMRefreshPeriod(1)
```

16.2.5 updateIRMRefreshPeriod

Online command that updates an existing refresh period.

16.2.5.1 Description

This command updates an existing refresh period. A refresh period is the maximum length of time that a user can use rights before the rights are refreshed from the server.

16.2.5.2 Syntax

```
updateIRMRefreshPeriod(pindex,duration,dtype)
```

Argument	Definition
pindex	Specifies the index of the refresh period.
duration	Specifies the value of the refresh period. Integer.
dtype	Specifies the unit of the refresh period. 'MINUTES', 'HOURS', 'DAYS', 'MONTHS' or 'YEARS'.

16.2.5.3 Examples

The following example updates the refresh period at index zero to have a duration of 5 hours:

```
wls:/base_domain/serverConfig> updateIRMRefreshPeriod(0,5,\"HOURS\")
```

The following example updates the refresh period at index zero to have a duration of 50 minutes:

```
wls:/base_domain/serverConfig> updateIRMRefreshPeriod(0,50,\"MINUTES\")
```

16.2.6 addIRMSyncWindow

Online command that creates a sync window.

16.2.6.1 Description

This command creates a sync window. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.6.2 Syntax

```
addIRMSyncWindow(day,stHrs,stMins,endHrs,endMins)
```

Argument	Definition
day	Specifies the weekday. 'MONDAY', 'TUESDAY', etc.
stHrs	Specifies the start hours. Integer.
stMins	Specifies the start minutes. Integer.
endHrs	Specifies the end hours. Integer.
endMins	Specifies the end minutes. Integer.

16.2.6.3 Example

The following example creates a sync window that will result in Oracle IRM Desktop attempting to contact the server between 9.30am and 6.30pm on Mondays:

```
wls:/base_domain/serverConfig> addIRMSyncWindow(\"MONDAY\",9,30,6,30)
```

16.2.7 getIRMSyncWindow

Online command that displays an existing sync window.

16.2.7.1 Description

This command displays a sync window that is present at the specified index. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.7.2 Syntax

```
getIRMSyncWindow(sindex)
```

Argument	Definition
sindex	Specifies the index of the sync window.

16.2.7.3 Examples

The following example displays the sync window at index zero:

```
wls:/base_domain/serverConfig> getIRMSyncWindow(0)
```

The following example displays the sync window at index one:

```
wls:/base_domain/serverConfig> getIRMSyncWindow(1)
```

16.2.8 getIRMSyncWindows

Online command that displays all existing sync windows.

16.2.8.1 Description

This command displays all existing sync windows. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.8.2 Syntax

```
getIRMSyncWindows()
```

16.2.8.3 Example

```
wls:/base_domain/serverConfig> getIRMSyncWindows()
```

16.2.9 removeIRMSyncWindow

Online command that removes an existing sync window.

16.2.9.1 Description

This command removes a sync window that is present at the specified index. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.9.2 Syntax

```
removeIRMSyncWindow(sindex)
```

Argument	Definition
sindex	Specifies the index of the sync window.

16.2.9.3 Examples

The following example removes the sync window at index zero:

```
wls:/base_domain/serverConfig> removeIRMSyncWindow(0)
```

The following example removes the sync window at index one:

```
wls:/base_domain/serverConfig> removeIRMSyncWindow(1)
```

16.2.10 updateIRMSyncWindow

Online command that updates an existing sync window.

16.2.10.1 Description

This command updates an existing sync window. A sync window is a period during which Oracle IRM Desktop will attempt to contact the server to synchronize rights.

16.2.10.2 Syntax

```
updateIRMSyncWindow(indexOfDay,day,stHrs,stMins,endHrs,endMins)
```

Argument	Definition
indexOfDay	Specifies the index of the sync window. Integer.
day	Specifies the weekday. 'MONDAY', 'TUESDAY', etc.
stHrs	Specifies the start hours. Integer.
stMins	Specifies the start minutes. Integer.

Argument	Definition
endHrs	Specifies the end hours. Integer.
endMins	Specifies the end minutes. Integer.

16.2.10.3 Example

The following example updates the sync window at index zero so that Oracle IRM Desktop will attempt to contact the server between 9.30am and 5.30pm on Mondays:

```
wls:/base_domain/serverConfig> updateIRMSyncWindow(0,\"MONDAY\",9,30,5,30)
```

16.2.11 getIRMCryptoSchema

Online command that displays the cryptography algorithm.

16.2.11.1 Description

This command displays the cryptography algorithm currently applied to files that are sealed using Oracle IRM.

16.2.11.2 Syntax

```
getIRMCryptoSchema()
```

16.2.11.3 Example

```
wls:/base_domain/serverConfig> getIRMCryptoSchema()
```

16.2.12 setIRMCryptoSchema

Online command that sets the cryptography algorithm.

16.2.12.1 Description

This command sets the cryptography algorithm that will be applied to files that are sealed using Oracle IRM. The default of AES128 is recommended.

16.2.12.2 Syntax

```
setIRMCryptoSchema(cryptID)
```

Argument	Definition
cryptID	Specifies the name of the cryptography algorithm. Possible algorithm names are AES128, AES256, AES128-FIPS, AES256-FIPS, DES3-FIPS.

16.2.12.3 Example

The following example sets the cryptography algorithm used for Oracle IRM communications to AES128:

```
wls:/base_domain/serverConfig> setIRMCryptoSchema(\"AES128\")
```

16.2.13 getIRMDeviceCount

Online command that displays the device count.

16.2.13.1 Description

This command displays the maximum number of devices on which a user can open a sealed document at one time. The value applies to all users, and does not differ for individual users.

16.2.13.2 Syntax

```
getIRMDeviceCount()
```

16.2.13.3 Example

```
wls:/base_domain/serverConfig> getIRMDeviceCount()
```

16.2.14 setIRMDeviceCount

Online command that sets the device count.

16.2.14.1 Description

This command sets the maximum number of devices on which a user can open a sealed document at one time. The value applies to all users. The device count is normally kept low (1 or 2) to make it difficult to circumvent document access restrictions by sharing passwords.

16.2.14.2 Syntax

```
setIRMDeviceCount(devCount)
```

Argument	Definition
devCount	Specifies the device count value (the number of devices). Integer.

16.2.14.3 Example

The following example sets the device count to 2:

```
wls:/base_domain/serverConfig> setIRMDeviceCount(2)
```

16.2.15 getIRMJournalCleanUp

Online command that displays the current report record clean-up values.

16.2.15.1 Description

This command displays the report record clean-up values. The values show how often report record clean-ups are performed, and the maximum age of the report records before they are deleted.

16.2.15.2 Syntax

```
getIRMJournalCleanUp()
```

16.2.15.3 Example

```
wls:/base_domain/serverConfig> getIRMJournalCleanUp()
```

16.2.16 setIRMJournalCleanUp

Online command that sets report record clean-up values.

16.2.16.1 Description

This command sets how often report record clean-ups are performed, and the maximum age of report records before they are deleted.

16.2.16.2 Syntax

```
setIRMJournalCleanUp(clDuration,clUnitType,retDuration,retUnitType)
```

Argument	Definition
clDuration	Specifies the value for how often report record clean-ups are performed. Integer.
clUnitType	Specifies the unit for how often report record clean-ups are performed. 'MINUTES', 'HOURS', 'DAYS', 'MONTHS', 'YEARS'.
retDuration	Specifies the value for the maximum age of report records before they are deleted. Integer.
retUnitType	Specifies the unit for the maximum age of report records before they are deleted. 'MINUTES', 'HOURS', 'DAYS', 'MONTHS', 'YEARS'.

16.2.16.3 Example

The following example runs report record clean-ups every 5 days, and deletes report records that are 6 months old:

```
wls:/base_domain/serverConfig> setIRMJournalCleanUp(5,\"DAYS\",6,\"MONTHS\")
```

16.2.17 getIRMLicenseStateCleanUp

Online command that displays the license state clean-up frequency.

16.2.17.1 Description

This command displays the license state clean-up frequency (the frequency at which license records will be deleted).

16.2.17.2 Syntax

```
getIRMLicenseStateCleanUp()
```

16.2.17.3 Example

```
wls:/base_domain/serverConfig> getIRMLicenseStateCleanUp()
```

16.2.18 setIRMLicenseStateCleanUp

Online command that sets the license state clean-up frequency.

16.2.18.1 Description

This command sets the license state clean-up frequency (the frequency at which license records will be deleted).

16.2.18.2 Syntax

```
setIRMLicenseStateCleanUp(duration,unitType)
```

Argument	Definition
duration	Specifies the value of the frequency at which license records will be deleted. Integer.
unitType	Specifies the unit for the frequency at which license records will be deleted. 'MINUTES', 'HOURS', 'DAYS', 'MONTHS', 'YEARS'.

16.2.18.3 Examples

The following example sets the frequency at which license records will be deleted to 10 hours:

```
wls:/base_domain/serverConfig> setIRMLicenseStateCleanUp(10,\"HOURS\")
```

The following example sets the frequency at which license records will be deleted to 50 minutes:

```
wls:/base_domain/serverConfig> setIRMLicenseStateCleanUp(50,\"MINUTES\")
```

16.2.19 getIRMPrivacyURL

Online command that displays the URL of the privacy statement page.

16.2.19.1 Description

This command displays the URL of the privacy statement page. The privacy statement page displays a statement that users must accept before viewing sealed content.

16.2.19.2 Syntax

```
getIRMPrivacyURL()
```

16.2.19.3 Examples

```
wls:/base_domain/serverConfig> getIRMPrivacyURL()
```

16.2.20 setIRMPrivacyURL

Online command that sets the URL of the privacy statement page.

16.2.20.1 Description

This command sets the URL of a privacy statement that users must accept before viewing sealed content.

16.2.20.2 Syntax

```
setIRMPrivacyURL(privacyURL)
```

Argument	Definition
privacyURL	Specifies the URL of the privacy statement page.

16.2.20.3 Example

The following example sets the URL of the privacy policy page to "http://irm.example.com/":

```
wls:/base_domain/serverConfig> setIRMPrivacyURL(\"http://irm.example.com/\")
```

16.2.21 getIRMKeyStore

Online command that displays the type and location of the Oracle IRM keystore.

16.2.21.1 Description

This command displays the type and location of the Oracle IRM keystore.

16.2.21.2 Syntax

```
getIRMKeyStore()
```

16.2.21.3 Examples

```
wls:/base_domain/serverConfig> getIRMKeyStore()
```

16.2.22 setIRMKeyStore

Online command that sets the type and location of the Oracle IRM keystore.

16.2.22.1 Description

This command sets the type and location of the Oracle IRM keystore. You should not normally need to change the keystore type and location from the default (type JKS at location ${domain.home}/config/fmwconfig/irm.jks).

16.2.22.2 Syntax

```
setIRMKeyStore()
```

You will be prompted to provide the following arguments:

Argument	Definition
KeyStore Type	Specifies the type of the keystore.
KeyStore Location	Specifies the location of the keystore.

16.2.22.3 Example

The following example sets the keystore type to JCEKS and the keystore location to D:/exampledir/:

```
wls:/base_domain/serverConfig> setIRMKeyStore()
Enter KeyStore Type: JCEKS
Enter KeyStore Location: D:/exampledir/
```

16.3 Migration Commands

Use the WLST commands listed in Table 16–3 to set up import and export of user data between instances of Oracle IRM Server.

Table 16–3 WLST Commands for Import and Export of Oracle IRM user data

Use this command...	To...	Use with WLST...
setIRMExportFolder	Set or clear the data export folder location.	Online
getIRMExportFolder	Display the value for the data export folder.	Online
setIRMImportFolder	Set or clear the data import folder location.	Online

Table 16–3 (Cont.) WLST Commands for Import and Export of Oracle IRM user data

Use this command...	To...	Use with WLST...
getIRMImportFolder	Display the value for the data import folder.	Online

16.3.1 setIRMExportFolder

Online command that sets or clears the data export folder location.

16.3.1.1 Description

This command sets or clears the location of the folder used for data export.

16.3.1.2 Syntax

```
setIRMExportFolder(folder)
```

Argument	Definition
folder	Specifies the data export folder value.

16.3.1.3 Example

```
wls:/base_domain/serverConfig> setIRMExportFolder("export")
```

16.3.2 getIRMExportFolder

Online command that displays the value of the data export folder.

16.3.2.1 Description

This command displays the location of the folder used for data export.

16.3.2.2 Syntax

```
getIRMExportFolder()
```

16.3.2.3 Example

```
wls:/base_domain/serverConfig> getIRMExportFolder()
```

16.3.3 setIRMImportFolder

Online command that sets or clears the data import folder location.

16.3.3.1 Description

This command sets or clears the location of the folder used for data import.

16.3.3.2 Syntax

```
setIRMImportFolder(folder)
```

Argument	Definition
folder	Specifies the import folder value.

16.3.3.3 Example

```
wls:/base_domain/serverConfig> setIRMImportFolder("import")
```

16.3.4 getIRMImportFolder

Online command that displays the value of the data import folder.

16.3.4.1 Description

This command displays the location of the folder used for data import.

16.3.4.2 Syntax

```
getIRMImportFolder()
```

16.3.4.3 Example

```
wls:/base_domain/serverConfig> getIRMImportFolder()
```

16.4 Test Content Commands

Use the WLST commands listed in Table 16–4 to set up test content for users of Oracle IRM Desktop.

Table 16–4 WLST Commands for Test Content

Use this command...	To...	Use with WLST...
addIRMTestContent	Create a new test content instance.	Online
getIRMTestContent	Display details for an existing test content instance.	Online
getIRMTestContents	Display details of all existing test content instances.	Online
removeIRMTestContent	Remove an existing test content instance.	Online
updateIRMTestContent	Update an existing text content instance.	Online

16.4.1 addIRMTestContent

Online command that creates a new test content instance.

16.4.1.1 Description

This command creates a test content instance. Test content instances identify an item of test content, usually an image file. Test content is shown in a sealed document when Oracle IRM Desktop successfully connects to Oracle IRM Server through the client test facility.

16.4.1.2 Syntax

```
addIRMTestContent(uri,localeKeys,testNames)
```

Argument	Definition
uri	Specifies the URI of the test content (for example, an image file).

Argument	Definition
localeKeys	Specifies the locale(s) associated with this test content instance. Must be from the list of two-letter language codes given in Table 16–5 (for example, 'fr' for French). If there is more than one supported locale for an instance, the two-letter codes must be listed as comma-separated values.
testNames	Specifies the name(s) associated with this test content instance. If there is more than one name for a URI, they must be specified as comma-separated values.

Table 16–5 Language codes (ISO 639-1 "two-letter codes")

Language/Code	Language/Code	Language/Code
Arabic: ar	Greek: el	Romanian: ro
Brazilian Portuguese: pt-BR	Hebrew: iw	Russian: ru
Czech: cs	Hungarian: hu	Simplified Chinese: zh-CN
Danish: da	Italian: it	Slovak: sk
Dutch: nl	Japanese: ja	Spanish: es
English: en	Korean: ko	Swedish: sv
Finnish: fi	Norwegian: no	Thai: th
French: fr	Polish: pl	Traditional Chinese: zh-TW
German: de	Portuguese: pt	Turkish: tr

16.4.1.3 Examples

The following example creates a test content instance comprising an image named exampleImage.jpg at http://irm.example.com, for use with English installations, and showing the name 'Test Content':

```
wls:/base_domain/serverConfig>
addIRMTestContent(\"http://irm.example.com/exampleImage.jpg\",\"en\",\"Test
Content\")
```

The following example creates a test content instance comprising an image named exampleImage.jpg at http://irm.example.com, for use with English and French installations, and showing the names 'Test Content (en)' and 'Test Content (fr)':

```
wls:/base_domain/serverConfig>
addIRMTestContent(\"http://irm.example.com/exampleImage.jpg\",\"en,fr\",\"Test
Content (en),Test Content (fr)\")
```

16.4.2 getIRMTestContent

Online command that displays the details of an existing test content instance.

16.4.2.1 Description

This command displays the details of the test content instance that is present at the specified index. Test content instances identify an item of test content, usually an image file. Test content is shown in a sealed document when Oracle IRM Desktop successfully connects to Oracle IRM Server through the client test facility.

16.4.2.2 Syntax

```
getIRMTestContent(tindex)
```

Argument	Definition
tindex	Specifies the index of the test content instance.

16.4.2.3 Examples

The following example displays the details of the test content instance at index zero:

```
wls:/base_domain/serverConfig> getIRMTestContent(0)
```

The following example displays the details of the test content instance at index one:

```
wls:/base_domain/serverConfig> getIRMTestContent(1)
```

16.4.3 getIRMTestContents

Online command that displays all the test content instances.

16.4.3.1 Description

This command displays all the test content instances. Test content instances identify an item of test content, usually an image file. Test content is shown in a sealed document when Oracle IRM Desktop successfully connects to Oracle IRM Server through the client test page.

16.4.3.2 Syntax

```
getIRMTestContents()
```

16.4.3.3 Example

```
wls:/base_domain/serverConfig> getIRMTestContents()
```

16.4.4 removeIRMTestContent

Online command that removes an existing test content instance.

16.4.4.1 Description

This command removes the test content instance that is present at the specified index. Test content instances identify an item of test content, usually an image file.

16.4.4.2 Syntax

```
removeIRMTestContent(tindex)
```

Argument	Definition
tindex	Specifies the index of test content.

16.4.4.3 Examples

The following example removes the test content instance at index zero:

```
wls:/base_domain/serverConfig> removeIRMTestContent(0)
```

The following example removes the test content instance at index one:

```
wls:/base_domain/serverConfig> removeIRMTestContent(1)
```

16.4.5 updateIRMTestContent

Online command that updates an existing test content instance.

16.4.5.1 Description

This command updates an existing test content instance. Test content instances identify an item of test content, usually an image file. Test content is shown in a sealed document when Oracle IRM Desktop successfully connects to Oracle IRM Server through the client test facility.

16.4.5.2 Syntax

```
updateIRMTestContent(tindex,uri,localeKeys,testNames)
```

Argument	Definition
tindex	Specifies the index of the test content instance. Integer.
uri	Specifies the URI of the test content (for example, an image file).
localeKeys	Specifies the locale(s) associated with this test content instance. Must be from the list of two-letter language codes given in Table 16–5 (for example, 'fr' for French). If there is more than one supported locale for an instance, the two-letter codes must be listed as comma-separated values.
testNames	Specifies the name(s) associated with this test content instance. If there is more than one name for a URI, they must be specified as comma-separated values.

16.4.5.3 Examples

The following example updates a test content instance by changing the image to exampleImage.jpg at http://irm.example.com, for use with English installations, and showing the name 'Test Content':

```
wls:/base_domain/serverConfig>
updateIRMTestContent(0,\"http://irm.example.com/exampleImage.jpg\",\"en\",\"Test
Content\")
```

The following example updates a test content instance by changing the image to exampleImage.jpg at http://irm.example.com, for use with English and French installations, and showing the names 'Test Content (English)' and 'Test Content (French)':

```
wls:/base_domain/serverConfig> updateIRMTestContent(0,
\"http://irm.example.com/exampleImage.jpg\",\"en,fr\",\"Test Content
(English),Test Content (French)\")
```

16.5 Languages Support Commands

Use the WLST commands listed in Table 16–6 to set up languages support for users of Oracle IRM Server.

Table 16–6 WLST Commands for Oracle IRM Server languages support

Use this command...	To...	Use with WLST...
addIRMTranslation	Create a new language support instance.	Online
getIRMDefaultTranslation	Display the default language.	Online
getIRMTranslations	Display all the language support instances.	Online
removeIRMTranslation	Remove an existing language support instance.	Online
setIRMTranslations	Set the default language, and set a language support instance for one or more additional languages.	Online

16.5.1 addIRMTranslation

Online command that creates a new language support instance.

16.5.1.1 Description

This command creates a new language support instance. Each language support instance provides the facility in Oracle IRM Server to add names and descriptions in one or more languages (in addition to the default language).

16.5.1.2 Syntax

```
addIRMTranslation(transList)
```

Argument	Definition
transList	Specifies the supported language(s). Must be from the list of two-letter language codes given in Table 16–5 (for example, 'fr' for French). If there is more than one supported language for an instance, the two-letter codes must be listed as comma-separated values.

16.5.1.3 Examples

The following example creates a language support instance that will enable users of Oracle IRM Server to add names and descriptions in French (in addition to their default language):

```
wls:/base_domain/serverConfig> addIRMTranslation(\"fr\")
```

The following example creates a language support instance that will enable users of Oracle IRM Server to add names and descriptions in French and Arabic (in addition to their default language):

```
wls:/base_domain/serverConfig> addIRMTranslation(\"fr,ar\")
```

16.5.2 getIRMDefaultTranslation

Online command that displays the default language.

16.5.2.1 Description

This command displays the default language.

16.5.2.2 Syntax

```
getIRMDefaultTranslation()
```

16.5.2.3 Example

```
wls:/base_domain/serverConfig> getIRMDefaultTranslation()
```

16.5.3 getIRMTranslations

Online command that displays all the language support instances.

16.5.3.1 Description

This command displays all the language support instances. Each language support instance provides the facility in Oracle IRM Server to add names and descriptions in one or more languages (in addition to the default language).

16.5.3.2 Syntax

```
getIRMTranslations()
```

16.5.3.3 Example

```
wls:/base_domain/serverConfig> getIRMTranslations()
```

16.5.4 removeIRMTranslation

Online command that removes an existing language support instance.

16.5.4.1 Description

This command removes the language support instance that is present at the specified index. Each language support instance provides the facility in Oracle IRM Server to add names and descriptions in one or more languages (in addition to the default language).

16.5.4.2 Syntax

```
removeIRMTranslation(tindex)
```

Argument	Definition
tindex	Specifies the index of the language support instance.

16.5.4.3 Examples

The following example removes the language support instance at index zero:

```
wls:/base_domain/serverConfig> removeIRMTranslation(0)
```

The following example removes the language support instance at index one:

```
wls:/base_domain/serverConfig> removeIRMTranslation(1)
```

16.5.5 setIRMTranslations

Online command that sets the default language, and sets a language support instance for one or more languages in addition to the default language.

16.5.5.1 Description

This command sets the default language, and sets a language support instance for one or more languages in addition to the default language. Each language support instance provides the facility in Oracle IRM Server to add names and descriptions in one or more languages (in addition to the default language).

16.5.5.2 Syntax

```
setIRMTranslations(defaultTrans,transList)
```

Argument	Definition
defaultTrans	Specifies the default language. Language code (for example, 'en' for English).
transList	Specifies the supported language(s). Must be from the list of two-letter language codes given in Table 16–5 (for example, 'fr' for French). If there is more than one supported language for an instance, the two-letter codes must be listed as comma-separated values.

16.5.5.3 Examples

The following example enables users of Oracle IRM Server to enter names and descriptions in English as the default language, and additionally to enter names and descriptions in French:

```
wls:/base_domain/serverConfig> setIRMTranslations(\"en\",\"fr\")
```

The following example enables users of Oracle IRM Server to enter names and descriptions in English as the default language, and additionally to enter names and descriptions in French and Arabic:

```
wls:/base_domain/serverConfig> setIRMTranslations(\"en\",\"fr,ar\")
```

16.6 Oracle IRM Desktop Installers Commands

Use the WLST commands listed in Table 16–7 to set up installation support for Oracle IRM Desktop software.

Table 16–7 WLST Oracle IRM Desktop Installers Commands

Use this command...	To...	Use with WLST...
addIRMDownload	Create a new installer.	Online
getIRMDownload	Display the details for an existing installer.	Online
getIRMDownloads	Display the details for all installers.	Online
removeIRMDownload	Remove an existing installer.	Online
updateIRMDownload	Update an existing installer.	Online

16.6.1 addIRMDownload

Online command that creates a new installer.

16.6.1.1 Description

This command creates a new installer. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.1.2 Syntax

```
addIRMDownload(locale,name,version,uri)
```

Argument	Definition
locale	Specifies the locale of the installer. Must be from the list of two-letter language codes given in Table 16–5 (for example, 'en' for English).
name	Specifies the name for the installer.
version	Specifies the version of the installer. This is a label for the installer, and is not verified against the associated installation software.
uri	Specifies the URI of Oracle IRM Desktop installation software.

16.6.1.3 Example

The following example creates an installer for English language installation software at http://irm.example.com/, with the name 'Oracle IRM Desktop' and the version number 11.1.1.1.0.0 visible to users of Oracle IRM Server when they select this installer:

```
wls:/base_domain/serverConfig> addIRMDownload(\"en\",\"Oracle IRM
Desktop\",\"11.1.1.1.0.0\",\"http://irm.example.com/\")
```

16.6.2 getIRMDownload

Online command that displays the details for an existing installer.

16.6.2.1 Description

This command displays the details for an installer that is present at the specified index. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.2.2 Syntax

```
getIRMDownload(dindex)
```

Argument	Definition
dindex	Specifies the index of the download.

16.6.2.3 Examples

The following example displays the details for the installer at index zero:

```
wls:/base_domain/serverConfig> getIRMDownload(0)
```

The following example displays the details for the installer at index one:

```
wls:/base_domain/serverConfig> getIRMDownload(1)
```

16.6.3 getIRMDownloads

Online command that displays the details of all installers.

16.6.3.1 Description

This command displays the details of all installers. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.3.2 Syntax

```
getIRMDownloads()
```

16.6.3.3 Example

```
wls:/base_domain/serverConfig> getIRMDownloads()
```

16.6.4 removeIRMDownload

Online command that removes an existing installer.

16.6.4.1 Description

Removes the installer that is present at the specified index. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.4.2 Syntax

```
removeIRMDownload(dindex)
```

Argument	Definition
dindex	Specifies the index of the download.

16.6.4.3 Examples

The following example removes the installer at index zero:

```
wls:/base_domain/serverConfig> removeIRMDownload(0)
```

The following example removes the installer at index one:

```
wls:/base_domain/serverConfig> removeIRMDownload(1)
```

16.6.5 updateIRMDownload

Online command that updates an existing installer.

16.6.5.1 Description

This command updates an existing installer. Each installer identifies the locale and URI of software for installing Oracle IRM Desktop, and displays a name and version number that enables users of Oracle IRM Server to select the installer.

16.6.5.2 Syntax

```
updateIRMDownload(dindex,locale,name,version,uri)
```

Argument	Definition
dindex	Specifies the index of the installer. Integer.

Argument	Definition
locale	Specifies the locale of the download. Must be from the list of two-letter language codes given in Table 16–5 (for example, 'en' for English).
name	Specifies the name for the installer.
version	Specifies the version of the installer. This is a label for the installer, and is not verified against the associated installation software.
uri	Specifies the URI for the Oracle IRM Desktop installation software.

16.6.5.3 Example

The following example updates the installer for index zero. After the update, the installation software is English language and is located at http://irm.example.com/. The name 'Oracle IRM Desktop (English)' and the version number 11.1.1.1.0.0 will be visible to users of Oracle IRM Server when they select this installer.

```
wls:/base_domain/serverConfig> updateIRMDownload(0,\"en\",\"Oracle IRM Desktop
(English)\",\"11.1.1.1.0.0\",\"http://irm.example.com/\")
```

17

Oracle WebCenter: Imaging Custom WLST Commands

The following sections describe the WLST commands that are specific to Oracle WebCenter: Imaging. Topics include:

- Section 17.1, "Overview of Imaging WLST Command Categories"
- Section 17.2, "Diagnostic Commands"
- Section 17.3, "Imaging Configuration Commands"

17.1 Overview of Imaging WLST Command Categories

WLST commands specific to Imaging are divided into the following categories.

Table 17–1 Imaging WLST Command Categories

Command category	Description
Diagnostic Commands	Return workflow agent and other processing information.
Imaging Configuration Commands	Configure settings specific to Imaging and Process Management.

17.2 Diagnostic Commands

Use the Imaging WLST diagnostic commands, listed in table Table 17–2, to list and organize processing failures during workflow processes.

Table 17–2 Diagnostic Commands for Imaging

Use this command...	To...	Use with WLST...
clearIPMWorkflowFaults	Clear processing failures that occurred during workflow agent processing.	Online
listIPMWorkflowFaults	Provide details of processing failures that occurred during workflow agent processing.	Online
repairIPMWorkflowFaults	Repair processing failures that occurred during workflow agent processing.	Online
sumIPMWorkflowFaults	Count processing failures during workflow agent processing, grouped by choice of date, application ID, or batch ID.	Online
resetIpmDMSMetrics	Reset DMS metrics to zero.	Online

17.2.1 clearIPMWorkflowFaults

Command Category: Diagnostic Commands

Use with WLST: Online

17.2.1.1 Description

Clear processing failures that have occurred during workflow agent processing.

17.2.1.2 Syntax

```
clearIPMWorkflowFaults([startDate], [endDate], [appId], [batchId])
```

Argument	Definition
startDate	Optional. The start of the date range for which error details should be repaired, in yyyy-MM-dd format.
endDate	Optional. The end of the date range for which error details should be repaired, in yyyy-MM-dd format.
appId	Optional. The application ID for which error details should be repaired, in yyyy-MM-dd format.
batchId	Optional. The batch ID for which error details should be repaired.

17.2.1.3 Example

The following example clears the faults within the specified parameters.

```
clearIPMWorkflowFaults(startDate="2009-06-01", endDate="2009-06-02")
clearIPMWorkflowFaults(appId=3)
clearIPMWorkflowFaults(batchId=15)
clearIPMWorkflowFaults(startDate="2009-06-01", endDate="2009-06-02", appid=3)
```

17.2.2 listIPMWorkflowFaults

Command Category: Diagnostic Commands

Use with WLST: Online

17.2.2.1 Description

List details on processing failures that have occurred during workflow agent processing.

17.2.2.2 Syntax

```
listIPMWorkflowFaults([startDate], [endDate], [appId], [batchId])
```

Argument	Definition
startDate	Optional. The start of the date range for which error details should be repaired, in yyyy-MM-dd format.
endDate	Optional. The end of the date range for which error details should be repaired, in yyyy-MM-dd format.
appId	Optional. The application ID for which error details should be repaired.
batchId	Optional. The batch ID for which error details should be repaired.

17.2.2.3 Example

The following example lists the faults within the specified parameters.

```
listIPMWorkflowFaults(startDate="2009-06-01", endDate="2009-06-02")
listIPMWorkflowFaults(appId=3)
listIPMWorkflowFaults(batchId=15)
listIPMWorkflowFaults(startDate="2009-06-01", endDate="2009-06-02", appId=3)
```

17.2.3 repairIPMWorkflowFaults

Command Category: Diagnostic Commands

Use with WLST: Online

17.2.3.1 Description

Repair processing failures that have occurred during workflow agent processing.

17.2.3.2 Syntax

```
repairIPMWorkflowFaults([startDate], [endDate], [appId], [batchId])
```

Argument	Definition
startDate	Optional. The start of the date range for which error details should be repaired, in yyyy-MM-dd format.
endDate	Optional. The end of the date range for which error details should be repaired, in yyyy-MM-dd format.
appId	Optional. The application ID for which error details should be repaired.
batchId	Optional. The batch ID for which error details should be repaired.

17.2.3.3 Example

The following example repairs the faults within the specified parameters.

```
repairIPMWorkflowFaults(startDate="2009-06-01", endDate="2009-06-02")
repairIPMWorkflowFaults(appId=3)
repairIPMWorkflowFaults(batchId=15)
repairIPMWorkflowFaults(startDate="2009-06-01", endDate="2009-06-02", appid=3)
```

17.2.4 sumIPMWorkflowFaults

Command Category: Diagnostic Commands

Use with WLST: Online

17.2.4.1 Description

Provides a count of processing failures that have occurred during workflow agent processing. The results are grouped by date, application ID, or batch ID.

17.2.4.2 Syntax

```
sumIPMWorkflowFaults(group)
```

Argument	Definition
groupOption	Required. One of the following: ■ **DATE**: Returns fault counts grouped by date. ■ **APPID**: Returns fault counts grouped by application ID. ■ **BATCHID**: Returns fault counts grouped by batch ID.

17.2.4.3 Example

The following example returns all workflow faults grouped first by date, then by applications ID, then again grouped by batch ID.

```
sumIPMWorkflowFaults(group="DATE")
sumIPMWorkflowFaults(group="APPID")
sumIPMWorkflowFaults(group="BATCHID")
```

17.2.5 resetIpmDMSMetrics

Command Category: Diagnostic Commands

Use with WLST: Online

17.2.5.1 Description

Resets all Dynamic Monitoring Server (DMS) metrics associated with I/PM to zero. This is generally done if the administrator finds that historical performance data is skewing the current results.

17.2.5.2 Syntax

```
resetIpmDMSMetrics()
```

17.2.5.3 Example

The following example resets all DMS metrics to zero.

```
resetIpmDMSMetrics()
```

17.3 Imaging Configuration Commands

Use the Imaging configuration commands, listed in Table 17–3, to list and set configuration values specific to Imaging.

Table 17–3 Configuration Commands for Imaging

Use this command...	To...	Use with WLST...
createIPMConnection	Creates a new Imaging connection from a connection definition file.	Online
getIPMConfig	Get an Imaging configuration setting value, similar to navigating to the custom Imaging config mbean and using the standard WLST *set* command.	Online
grantIPMCredAccess	Grants CredentialAccessPermissions to Imaging when Imaging Managed Servers are in a separate domain home from the Administration Server.	Online
importIPMApplication	Imports an application definition from a previously exported definition file.	Online

Table 17–3 (Cont.) Configuration Commands for Imaging

Use this command...	To...	Use with WLST...
importIPMInput	Imports an input definition from a previously exported definition file.	Online
importIPMSearch	Imports a search definition from a previously exported definition file.	Online
listIPMConfig	Lists Imaging configuration mbeans.	Online
listIPMExportFile	Lists the contents of an exported Imaging definitions file.	Online
refreshIPMSecurity	Refresh security items currently stored in the Imaging database.	Online
setIPMConfig	Sets an Imaging configuration value.	Online
submitIPMToWorkflow	Submits a document to the workflow agent.	Online

17.3.1 createIPMConnection

Command Category: Imaging Configuration Commands

Use with WLST: Online

17.3.1.1 Description

Creates a new Imaging connection from a connection definition file. The connection definition file is an XML file that describes a single Imaging connection definition using the Connection element type from the Imaging ConnectionService web services API schema definition. This schema is available from a running Imaging server using at the following URL:

`http://`*`ipm_server_machine`*`:`*`ipm_server_port`*`/imaging/ws/ConnectionService?xsd=1`

For more information about the connection definition file format, see the *Oracle WebCenter Content Administrator's Guide for Imaging*.

17.3.1.2 Syntax

```
createIPMConnection(connectionFile)
```

Argument	Definition
connectionFile	Required. A full path to the connection definition file's location on the Imaging server Node. Must be enclosed in single or double quotes.

17.3.1.3 Example

The following example creates a connection based on the specified attribute.

```
createIPMConnection(connectionFile="/home/ipmuser/localCSConnection.xml")
```

17.3.2 getIPMConfig

Command Category: Imaging Configuration Commands

Use with WLST: Online

17.3.2.1 Description

Gets an Imaging configuration setting value. The command is equivalent to browsing the custom mbean hierarchy to the Imaging config mbean and using the standard WLST *set* command to set an mbean attribute.

17.3.2.2 Syntax

```
getIPMConfig(attrName)
```

Argument	Definition
attrName	Required. Name of the attribute to be read. Must be enclosed in single or double quotes.

17.3.2.3 Example

The following example returns the value for the specified attribute names.

```
getIPMConfig('AgentUser')
getIPMConfig('CheckInterval')
```

17.3.3 grantIPMCredAccess

Grants CredentialAccessPermissions to Imaging so that it can read credentials from the credential store. This command is required in configurations where Imaging managed servers are in a separate domain home from the Administration Server. When at least one Imaging managed server is in the same domain home as the Administration Server, this command is not required, as CredentialAccessPermissions are granted during Imaging startup.

When the Imaging Managed Server is not in the same domain home as the Administration Server, however, the Imaging startup grant only affects the local settings. Local settings get overwritten when the Administration Server synchronizes its copy as the domain wide configuration, so this command updates the Administration Server configuration such that permissions are distributed correctly to all domain nodes.

17.3.3.1 Syntax

```
grantIPMCredAccess()
```

17.3.3.2 Example

The following example returns a list of all Imaging configuration mbeans.

```
grantIPMCredAccess()
```

17.3.4 importIPMApplication

Imports an application definition from a previously exported definition file.

17.3.4.1 Syntax

```
importIPMApplication(exportFile, action, name, repository, securityOption,
securityMember, docSecurityOption, docSecurityGroup, storageOption, storageVolume)
```

Argument	Definition
exportFile	Required. A full path to the export definition file's location on the Imaging server node. Must be enclosed in single or double quotes.
action	Required. The action to be performed. Available actions are: ■ **Add**: Creates a new input. Fails if an application with the same name already exists. ■ **Update**: Modifies and existing input. Fails if an application with the same name does not exist. ■ **AddOrUpdate**: Creates a new application if it does not already exist or updates one that does.
name	Required. The name of the application being imported from the exported definitions file.
repository	The name of the repository in which to create the application. Required when adding an application, ignored when updating or modifying an application.
securityOption	Optional. Specifies how to define security for the imported application as follows: ■ **Existing**: Uses application security as defined in the existing definition. Valid only for an update action. ■ **Imported**: Attempts to use application security as defined in the import file. Fails if any members defined in the import file are invalid. ■ **ValidOnly**: Uses application security as defined in the import file and filters out any invalid members. ■ **CurrentUser**: Sets full permissions to the user used to connect to the server. ■ **User**: Sets full permissions to the user name provided in the securityMember parameter. ■ **Group**: Sets full permissions to the group name provided in the securityMember parameter.
securityMember	Name of the user or group given full permissions to the application. Valid only when securityOption is set to either *User* or *Group*, otherwise it is ignored.
docSecurityOption	Optional. Specifies how to define document security for the imported application. ■ **Existing**: Uses document security as defined in the existing application. Valid only for an update action. ■ **Imported**: Attempts to use document security as defined in the import file. Fails if any members defined in the import file are invalid. ■ **ValidOnly**: Uses document security as defined in the import file and filters out any invalid members. ■ **Group**: Sets full permissions to the group name provided in the docSecurityGroup parameter.
docSecurityGroup	Name of group given full permissions to document security. Valid only when docSecurityOption is set to *Group*, otherwise it is ignored.

Argument	Definition
storageOption	Optional. Specifies how to define the storage policy for the imported application. ■ **Existing**: Uses the document storage policy as defined in the existing application. Valid only for an update action. ■ **Imported**: Attempts to use storage policy as defined in the import file. ■ **Volume**: Uses the specific volume named in the storageVolume parameter. ■ **Default**: Sets up the storage policy to use the system default volume.
storageVolume	Required. Volume for setting storage policy. Valid only when a storageOption of *Volume* is used. Ignored otherwise.

17.3.4.2 Example

The following example updates an existing application named *Invoices*. Note that the repository is listed as **None** because the update action uses the repository specified in the original application.

```
importIPMApplication(exportFile="/home/ipmuser/exportdefinitions.xml",
action="Update", name="Invoices", repository=None, securityOption="Existing")
```

17.3.4.3 Example

The following example creates a new application named *Receipts*. Note that the repository is explicitly named because the add action requires a valid repository be named.

```
importIPMApplication(exportFile="/home/ipmuser/exportdefinitions.xml",
action="Add", name="Receipts", repository="LocalCS", securityOption="ValidOnly")
```

17.3.5 importIPMInput

Imports an input definition from a previously exported definition file.

17.3.5.1 Syntax

```
importIPMInput(exportFile, action, name, securityOption, securityMember)
)
```

Argument	Definition
exportFile	Required. A full path to the export definition file's location on the Imaging server node. Must be enclosed in single or double quotes.
action	Required. The action to be performed. Available actions are: ■ **Add**: Creates a new input. Fails if an input with the same name already exists. ■ **Update**: Modifies an existing input. Fails if an input with the same name does not exist. ■ **AddOrUpdate**: Creates a new application if it does not already exist or updates one that does.

Argument	Definition
name	Required. The name of the input being imported from the exported definitions file.
repository	The name of the repository in which to create the application. Required when adding an application, ignored when updating or modifying an application.
securityOption	Optional. Specifies how to define security for the imported application as follows: ■ **Existing**: Uses input security as defined in the existing definition. Valid only for an update action. ■ **Imported**: Attempts to use input security as defined in the import file. Fails if any members defined in the import file are invalid. ■ **ValidOnly**: Uses input security as defined in the import file and filters out any invalid members. ■ **CurrentUser**: Sets full permissions to the user used to connect to the server. ■ **User**: Sets full permissions to the user name provided in the securityMember parameter. ■ **Group**: Sets full permissions to the group name provided in the securityMember parameter.
securityMember	Name of the user or group given full permissions to the input. Valid only when securityOption is set to either *User* or *Group*, otherwise it is ignored.

17.3.5.2 Example

The following example updates an existing input named *Invoices*. Note that the repository is listed as **None** because the update action uses the repository specified in the original application.

```
importIPMInput(exportFile="/home/ipmuser/exportdefinitions.xml", action="Update",
name="Invoices", securityOption="Existing")
```

17.3.5.3 Example

The following example creates a new input named *Receipts*. Note that the repository is explicitly named because the add action requires a valid repository be named.

```
importIPMInput(exportFile="/home/ipmuser/exportdefinitions.xml", action="Add",
name="Receipts", securityOption="ValidOnly")
```

17.3.6 importIPMSearch

Import a search definition from a previously exported definition file.

17.3.6.1 Syntax

```
importIPMSearch(exportFile, action, name, securityOption, securityMember)
```

Argument	Definition
exportFile	Required. A full path to the export definition file's location on the Imaging server node. Must be enclosed in single or double quotes.

Argument	Definition
action	Required. The action to be performed. Available actions are: ■ **Add**: Creates a new search. Fails if a search with the same name already exists. ■ **Update**: Modifies an existing search. Fails if a search with the same name does not exist. ■ **AddOrUpdate**: Creates a new search if it does not already exist or updates one that does.
name	Required. The name of the search being imported from the exported definitions file.
repository	The name of the repository in which to create the application. Required when adding an application, ignored when updating or modifying an application.
securityOption	Optional. Specifies how to define security for the imported application as follows: ■ **Existing**: Uses search security as defined in the existing definition. Valid only for an update action. ■ **Imported**: Attempts to use search security as defined in the import file. Fails if any members defined in the import file are invalid. ■ **ValidOnly**: Uses search security as defined in the import file and filters out any invalid members. ■ **CurrentUser**: Sets full permissions to the user used to connect to the server. ■ **User**: Sets full permissions to the user name provided in the securityMember parameter. ■ **Group**: Sets full permissions to the group name provided in the securityMember parameter.
securityMember	Name of the user or group given full permissions to the search. Valid only when securityOption is set to either *User* or *Group*, otherwise it is ignored.

17.3.6.2 Example

The following example updates an existing search named *Invoices*. Note that the repository is listed as **None** because the update action uses the repository specified in the original application.

```
importIPMSearch(exportFile="/home/ipmuser/exportdefinitions.xml", action="Update",
name="Invoices", securityOption="Existing")
```

17.3.6.3 Example

The following example creates a new search named *Receipts*. Note that the repository is explicitly named because the add action requires a valid repository be named.

```
importIPMSearch(exportFile="/home/ipmuser/exportdefinitions.xml", action="Add",
name="Receipts", securityOption="ValidOnly")
```

17.3.7 listIPMConfig

Command Category: Imaging Configuration Commands

Use with WLST: Online

17.3.7.1 Description

Provides a listing of Imaging configuration mbeans. The command is equivalent to browsing the custom mbean hierarchy and listing the Imaging mbean attributes.

17.3.7.2 Syntax

```
listIPMConfig()
```

17.3.7.3 Example

The following example returns a list of all Imaging configuration mbeans.

```
listIPMConfig()
```

17.3.8 listIPMExportFile

Lists the contents of an exported Imaging definitions file.

17.3.8.1 Syntax

```
listIPMExportFile(exportFile="<path to file>")
```

Argument	Definition
exportFile	Required. A full path to the export definition file's location on the Imaging server node. Must be enclosed in single or double quotes.

17.3.8.2 Example

The following example returns the contents of an Imaging definitions file.

```
listIPMExportFile(exportFile="/home/ipmuser/exportdefinitions.xml")
```

17.3.9 refreshIPMSecurity

Command Category: Imaging Configuration Commands

Use with WLST: Online

17.3.9.1 Description

Refreshes security items currently stored in the Imaging database. This is typically done when migrating security to a different policy store and only updates security items found in the new policy store.

17.3.9.2 Syntax

```
refreshIPMSecurity()
```

17.3.9.3 Example

The following example refreshes the security items stored in the Imaging database.

```
refreshIPMSecurity()
```

17.3.10 setIPMConfig

Command Category: Imaging Configuration Commands

Use with WLST: Online

17.3.10.1 Description

Sets an Imaging configuration setting value. The command is equivalent to browsing the custom mbean hierarchy to the Imaging config mbean and using the standard WLST 'set' command to set an mbean attribute.

17.3.10.2 Syntax

```
setIPMConfig(attrName, value)
```

Argument	Definition
attrName	Required. Name of the attribute to be set. Must be enclosed in single or double quotes.
value	Required. Value of the attribute to be set. Only enclosed in single or double quotes if value is a string literal.

17.3.10.3 Example

The following example sets the specified values for the specified attribute names.

```
setIPMConfig('AgentUser', 'agentadmin')
setIPMConfig('CheckInterval', 30)
```

17.3.11 submitIPMToWorkflow

Submits a document to the workflow agent. Note that a confirmation message is displayed stating that the document has been submitted, however if the document is stored in an application that is not configured with a workflow, no action is taken.

17.3.11.1 Syntax

```
submitIPMToWorkflow(documentId)
```

Argument	Definition
documentId	Required. The unique document ID of the submitted document.

17.3.11.2 Example

The following example submits a document to a workflow.

```
submitIPMToWorkflow(documentId="2.IPM_12345")
```

18

Oracle Business Process Management Custom WLST Commands

This chapter lists and describes the custom WLST commands for Oracle Business Process Management.

18.1 BPMLifecycleAdmin Command Group

Table 18–1 lists and describes the BPMLifecycleAdmin commands for project lifecycle administration.

Table 18–1 BPMLifecycleAdmin Commands for Project Lifecycle Administration

Use this command...	To...	Use with WLST...
create_public_share	Create a public share	Offline
unlock_public_share	Unlock a public share	Offline
export_public_share	Export a public share to the file system	Offline
delete_public_share	Delete a public share	Offline
publish_template	Publish a template to MDS	Offline
export_template	Export a template to the file system	Offline
delete_template	Delete a template from MDS	Offline

18.1.1 create_public_share

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

18.1.1.1 Description

Use this command to create a public share from a template. The template must exist in MDS.

18.1.1.2 Syntax

```
create_public_share(composerUser, composerPassword, connectionURL, templateName,
publicshareId, mdsconfigLocation, [Override], [oracleHome] )
```

Argument	Definition
composerUser	The Business Process Composer user who performs the current operation.

Argument	Definition
composerPassword	BPM Composer user's password
connectionURL	JNDI connection URL to the security server service in format *host*:*port*
templateName	Name of the template in MDS
publicshareId	Name of the public share to be created
mdsconfigLocation	Location of the `mds-config.xml` to be used to connect to MDS
`projectLocation`	The path where the public share will be created. If the path does not exists it will be created. The root is '/'.
Override	Enables you to override the public share if a public share exists in MDS with the same name. The template is not overwritten when you execute this command.
oracleHome	Optional. The Oracle home to be used.

18.1.1.3 Examples

The following example creates a public share named Sample_PublicShare. It is based on the template with name `Sample_Template`. The name of the public share is `Sample_PublicShare`, and the location of the `mds-config.xml` file is `/tmp/mds-config.xml`.

```
create_public_share('user_name', 'password', 'host:port','Sample_Template',
'Sample_PublicShare','/tmp/mds-config.xml')
```

The following example creates a public share named `Sample_PublicShare`. It is based on the template named `Sample_Template` that exists in MDS. The public share, not the template, is overridden. The location of the `mds-config.xml` file is `/tmp/mds-config.xml`.

```
create_public_share('user_name', 'password', 'host:port','Sample_Template',
'Sample_PublicShare','/tmp/mds-config.xml','true')"
```

18.1.2 unlock_public_share

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

18.1.2.1 Description

Use this command to unlock a public share. For example, when you create project by using the Ant task `create_public_share` command, the project is created as locked. You can then unlock it by using the `unlock_public_share` command.

A lock is also set by enabling or disabling the check box **enable sharing** in the project creation page in Oracle Business Process Composer.

It is also released when the user publishes a project from Business Process Composer.

The public share must exist in MDS.

18.1.2.2 Syntax

```
unlock_public_share(composerUser, composerPassword, connectionURL, publicshareId,
mdsconfigLocation, [oracleHome] )
```

Argument	Definition
composerUser	The Business Process Composer user who performs the current operation.
composerPassword	BPM Composer user's password
connectionURL	JNDI connection URL to the security server service in format *host*:*port*
publicshareId	Name of the public share to be unlocked
mdsconfigLocation	Location of the `mds-config.xml` to be used to connect to MDS
oracleHome	Optional. The Oracle home to be used

18.1.2.3 Example

The following example unlocks a public share named `Sample_PublicShare`. The location of the `mds-config.xml` file is `/tmp/mds-config.xml`.

```
unlock_public_share('user_name', 'password', 'host:port','Sample_PublicShare',
'/tmp/mds-config.xml')
```

18.1.3 export_public_share

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

18.1.3.1 Description

Use this command to export the public share from MDS to the file system.

18.1.3.2 Syntax

```
export_public_share(composerUser, composerPassword, connectionURL,
publicshareId,fsLocation, mdsconfigLocation, [oracleHome] )
```

Argument	Definition
composerUser	The Business Process Composer user who performs the current operation.
composerPassword	BPM Composer user's password
connectionURL	JNDI connection URL to the security server service in format *host*:*port*
publicshareId	Name of the public share to be exported
fsLocation	File system location where the project is to be downloaded
mdsconfigLocation	Location of the `mds-config.xml` to be used to connect to MDS
oracleHome	Optional. The Oracle home to be used

18.1.3.3 Example

The following example specifies the public share name as Sample_PublicShare, the file system location as /tmp, and the location of the `mds-config.xml` file as `/tmp/mds-config.xml`.

```
export_public_share('user_name', 'password', 'host:port','Sample_
PublicShare','/tmp', '/tmp/mds-config.xml')
```

18.1.4 delete_public_share

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

18.1.4.1 Description

Use this command to delete a public share from MDS. Executing this command requires that the public share is not locked.

18.1.4.2 Syntax

```
delete_public_share(composerUser, composerPassword, connectionURL, publicshareId,
mdsconfigLocation, [releaseLock], [oracleHome] )
```

Argument	Definition
composerUser	The Business Process Composer user who performs the current operation.
composerPassword	BPM Composer user's password
connectionURL	JNDI connection URL to the security server service in format *host*:*port*
publicshareId	Name of the public share to be deleted
mdsconfigLocation	Location of the `mds-config.xml` to be used to connect to MDS
releaseLock	Optional. If the public share is locked, this lock can be released and the delete operation completed. You can set this attribute to either `true` or `false`. If not specified, default value is `false`.
oracleHome	Optional. The Oracle home to be used

18.1.4.3 Examples

The following example specifies the name and location of a public share to be deleted.

```
delete_public_share('Sample_PublicShare','/tmp/mds-config.xml')
```

The following example specifies the name and location of a public share to be deleted, and that the public share should be deleted even if locked.

```
delete_public_share('user_name', 'password', 'host:port','Sample_
PublicShare','/tmp/mds-config.xml','true')
```

18.1.5 publish_template

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

18.1.5.1 Description

Use this command to publish the template from the file system to MDS.

18.1.5.2 Syntax

```
publish_template(composerUser, composerPassword, connectionURL, templateName,
fsLocation, mdsconfigLocation, [Override], [oracleHome] )
```

Argument	Definition
composerUser	The Business Process Composer user who performs the current operation.
composerPassword	BPM Composer user's password
connectionURL	JNDI connection URL to the security server service in format *host*:*port*
templateName	Name of the template to be published
fsLocation	File system location of the template project
mdsconfigLocation	Location of the `mds-config.xml` to be used to connect to MDS
`projectLocation`	The path where the public share will be created. If the path does not exists it will be created. The root is '/' .
Override	When you publish a template in MDS, this attribute enables you to override an existing template with the same name. Can either be 'true' or 'false'. If not specified, default value is 'false'.
oracleHome	Optional. The Oracle home to be used

18.1.5.3 Example

The following example publishes a template named `Sample_Template_Name_MDS`. to the root folder.

```
f('user_name', 'password', 'host:port','Sample_Template','/tmp/MyTemplate','/',
'/tmp/mds-config.xml')
```

The following example publishes a template named `Sample_Template_Name_MDS`.to the `'/WorkingOn/'` folder.

```
publish_template('user_name', 'password', 'host:port','Sample_
Template','/tmp/MyTemplate','/WorkingOn', '/tmp/mds-config.xml')
```

18.1.6 export_template

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

18.1.6.1 Description

Use this command to export the template from MDS to the file system.

18.1.6.2 Syntax

```
export_template(composerUser, composerPassword, connectionURL, templateName,
fsLocation, mdsconfigLocation, [oracleHome] )
```

Argument	Definition
composerUser	The Business Process Composer user who performs the current operation.
composerPassword	BPM Composer user's password
connectionURL	JNDI connection URL to the security server service in format *host*:*port*
templateName	Name of the template to be exported

Argument	Definition
fsLocation	File system location where the project is to be downloaded
mdsconfigLocation	Location of the mds-config.xml to be used to connect to MDS
oracleHome	Optional. The Oracle home to be used

18.1.6.3 Example

The following example specifies the template name as Sample_Template, the file system location as /tmp, and the location of the mds-config.xml file as /tmp/mds-config.xml.

```
export_template('user_name', 'password', 'host:port','Sample_
Template','/tmp','/tmp/mds-config.xml')
```

18.1.7 delete_template

Command Category: BPMLifecycleAdmin Commands

Use with WLST: Offline

18.1.7.1 Description

Use this command to delete the template from MDS.

18.1.7.2 Syntax

```
delete_template(composerUser, composerPassword, connectionURL, templateName,
mdsconfigLocation, [oracleHome] )
```

Argument	Definition
composerUser	The Business Process Composer user who performs the current operation.
composerPassword	BPM Composer user's password
connectionURL	JNDI connection URL to the security server service in format *host*:*port*
templateName	Name of the template to be deleted
fsLocation	File system location of the template project
mdsconfigLocation	Location of the mds-config.xml to be used to connect to MDS
projectLocation	The path where the public share will be created. If the path does not exists it will be created. The root is '/' .
oracleHome	Optional. The Oracle home to be used

18.1.7.3 Example

The following example deletes the template named Sample_template from MDS.

```
delete_template('weblogic', 'welcome1', 'host:port',
'/Sample_template','/tmp/mds-config.xml')
```

19

Oracle WebCenter Content Custom WLST Commands

The following sections describe the custom WLST commands for Oracle WebCenter Content. These commands enable you to configure and monitor the WebCenter Content server and the Content Server instance from the command line. Topics include:

For additional information about WebCenter Content and Content Server administration and configuration, see *Oracle WebCenter Content System Administrator's Guide for Content Server*.

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

19.1 Overview of WLST WebCenter Content Command Categories

WLST WebCenter Content commands are divided into the following categories:

Table 19–1 WLST WebCenter Content Command Categories

Command Category	Description
Server Configuration Commands	View and manage configuration for the Content Server instance.
E-Mail Configuration Commands	View and manage configuration for Content Server e-mail.
Additional Commands	View status information for the Content Server instance.

19.2 WLST WebCenter Content Help

To view the WebCenter Content commands that can be invoked from WLST, enter the following command at the WLST prompt:

```
help('UCM')
```

To view help for a specific WebCenter Content command, replace the `'UCM'` with the name of the command; for example:

```
help('getUCMServerPort')
```

19.3 Getter and Setter Methods Implementation

The WLST component for WebCenter Content uses **getter** and **setter** methods to handle a situation where multiple applications register their corresponding Mbeans on a managed server, but WLST can talk to only one application.

Getter Method

The **getter** method is designed to handle zero or one argument.

If you do not provide an argument to an WLST WebCenter Content command, then one of two things occurs:

- If only one application has registered its Mbean on the server, then the WLST WebCenter Content command should work successfully and display the output.
- If multiple applications have registered Mbeans on the server, then an error message is displayed to prompt you to enter the specific application name in the argument.

If there is one argument to an WLST WebCenter Content command, then the following occurs:

- You must enter the correct application name when entering an argument. If the name is not entered properly, then an error message is displayed to prompt you to enter the valid application name in the argument.

Setter Method

The **setter** method is designed to handle one or two arguments.

- The first argument is the *value* to which you want to set the parameter.
- The second argument is the *application name*, which can be null or a string.

19.4 Server Configuration Commands

Use the commands in Table 19–2 to configure the Oracle WebCenter Content Server instance.

Table 19–2 WLST Server Configuration Commands

Use this command...	To...	Use with WLST...
getUCMHttpServerAddress	Display the HTTP Server Address value.	Online
setUCMHttpServerAddress	Set the HTTP Server Address value.	Online
getUCMServerPort	Display the Intradoc Server Port configuration parameter.	Online
setUCMServerPort	Set the Intradoc Server Port configuration parameter.	Online
getUCMIPAddressFilter	Display the IP Address Filter value.	Online
setUCMIPAddressFilter	Set the IP Address Filter value.	Online

Table 19–2 (Cont.) WLST Server Configuration Commands

Use this command...	To...	Use with WLST...
getUCMUseSSL	Display the Use SSL value.	Online
setUCMUseSSL	Set the Use SSL value.	Online

19.4.1 getUCMHttpServerAddress

Use with WLST: Online

19.4.1.1 Description

Gets the HTTP Server Address value from the config.cfg file and displays it.

19.4.1.2 Syntax

```
getUCMHttpServerAddress()
```

or

```
getUCMHttpServerAddress(application_name)
```

19.4.1.3 Example

The following command displays the WebCenter Content HTTP server address for the application "Content Server":

```
getUCCHttpServerAddress('Content Server')
server.mycompany.com
```

19.4.2 setUCMHttpServerAddress

Use with WLST: Online

19.4.2.1 Description

Sets the HTTP Server Address value in the config.cfg file. The HTTP Server Address can be of the form *abc.xyz.def* or an IP address with port number.

The HTTP Server Address is used to formulate full URLs in the Content Server user interface.

19.4.2.2 Syntax

```
setUCMHttpServerAddress()
```

or

```
setUCMHttpServerAddress(value,application_name)
```

19.4.2.3 Example

The following command sets the Oracle WebCenter Content HTTP server address for the application "Content Server":

```
setUCMHttpServerAddress(server.mycompany.com,'Content Server')
```

19.4.3 getUCMServerPort

Use with WLST: Online

19.4.3.1 Description

Gets the Intradoc Server Port configuration parameter from the config.cfg file and displays it.

19.4.3.2 Syntax

```
getUCMServerPort()
```

or

```
getUCMServerPort(application_name)
```

19.4.3.3 Example

The following command displays the Intradoc Server Port value for the application "Content Server":

```
getUCMServerPort('Content Server')
4442
```

19.4.4 setUCMServerPort

Use with WLST: Online

19.4.4.1 Description

Sets the Server Port configuration parameter. The Server Port must be a positive integer between 0 and 65535.

19.4.4.2 Syntax

```
setUCMServerPort(value)
```

or

```
setUCMServerPort(value,application_name)
```

19.4.4.3 Example

The following command sets the Server Port configuration parameter for the application "Content Server":

```
setUCMServerPort(4442,'Content Server')
```

19.4.5 getUCMIPAddressFilter

Use with WLST: Online

19.4.5.1 Description

Gets the IP Address Filter value from the config.cfg file and displays it.

19.4.5.2 Syntax

```
getUCMIPAddressFilter()
```

or

```
getUCMIPAddressFilter(application_name)
```

19.4.5.3 Example

The following command displays the IP address filter value for the application "Content Server":

```
getUCMIPAddressFilter('Content Server')
10.131.123.*
```

19.4.6 setUCMIPAddressFilter

Use with WLST: Online

19.4.6.1 Description

Sets the WebCenter Content IP Address Filter value, which must be of "*.*.*.*" format or IPV6 Format. The value must be taken from a list of IP Addresses allowed to communicate with the Content Server instance through the Intradoc Server Port.

19.4.6.2 Syntax

`setUCMIPAddressFilter(`*`value`*`)`

or

`setUCMIPAddressFilter(`*`value,application_name`*`)`

19.4.6.3 Example

The following command sets the value for the WebCenter Content IP address filter for the application "Content Server":

```
setUCMIPAddressFilter(10.131.123.*,'Content Server')
```

19.4.7 getUCMUseSSL

Use with WLST: Online

19.4.7.1 Description

Gets the Use SSL value from the config.cfg file and displays it. The value can be `True` or `False`.

19.4.7.2 Syntax

`getUCMUseSSL()`

or

`getUCMUseSSL(`*`application_name`*`)`

19.4.7.3 Example

The following command displays the Use SSL value for the application "Content Server":

```
getUCMUseSSL('Content Server')
True
```

19.4.8 setUCMUseSSL

Use with WLST: Online

19.4.8.1 Description

Sets the Use SSL value in the config.cfg file. The value can be `True` or `False`.

19.4.8.2 Syntax

`setUCMUseSSL(`*`value`*`)`

or

`setUCMUseSSL(`*`value,application_name`*`)`

19.4.8.3 Example

The following command sets the Use SSL value for the application "Content Server":

```
setUCMUseSSL(True,'Content Server')
```

19.5 E-Mail Configuration Commands

Use the commands in Table 19–3 to configure e-mail for the Oracle WebCenter Content Server instance.

Table 19–3 WLST E-Mail Configuration Commands

Use this command...	To...	Use with WLST...
getUCMMailServer	Display the Mail Server value.	Online
setUCMMailServer	Set the Mail Server value.	Online
getUCMSmtpPort	Display the SMTP Port value.	Online
setUCMSmtpPort	Set the SMTP Port value.	Online
getUCMSysAdminAddress	Display the Admin Address value.	Online
setUCMSysAdminAddress	Set the Admin Address value.	Online

19.5.1 getUCMMailServer

Use with WLST: Online

19.5.1.1 Description

Gets the Mail Server value from the config.cfg file and displays it.

19.5.1.2 Syntax

`getUCMMailServer()`

or

`getUCMMailServer(`*`application_name`*`)`

19.5.1.3 Example

The following command displays the Mail Server value for the application "Content Server":

```
getUCMMailServer('Content Server')
mymailserver.mycompany.com
```

19.5.2 setUCMMailServer

Use with WLST: Online

19.5.2.1 Description

Sets the Mail Server value in the config.cfg file. The Mail Server value is the name of the mail server that the Content Server instance uses to send SMTP based e-mail.

19.5.2.2 Syntax

`setUCMMailServer(`*`value`*`)`

or

`setUCMMailServer(`*`value,application_name`*`)`

19.5.2.3 Example

The following command sets the value for the Mail Server for the application "Content Server":

```
setUCMMailServer(mymailserver.mycompany.com,'Content Server')
```

19.5.3 getUCMSmtpPort

Use with WLST: Online

19.5.3.1 Description

Gets the SMTP Port value in the config.cfg file and displays it.

19.5.3.2 Syntax

`getUCMSmtpPort()`

or

`getUCMSmtpPort(`*`application_name`*`)`

19.5.3.3 Example

The following command displays the SMTP port value for the application "Content Server":

```
getUCMSmtpPort('Content Server')
4055
```

19.5.4 setUCMSmtpPort

Use with WLST: Online

19.5.4.1 Description

Sets the SMTP Port value in the config.cfg file. The SMTP Port must be a positive integer between 1 and 65535. To reset the port to null, enter None for the value: setUCMSmtpPort(None)

19.5.4.2 Syntax

`setUCMSmtpPort(`*`value`*`)`

or

`setUCMSmtpPort(`*`value,application_name`*`)`

19.5.4.3 Example

The following command sets the SMTP port value for the application "Content Server":

```
setUCMSmtpPort(4055,'Content Server')
```

19.5.5 getUCMSysAdminAddress

Use with WLST: Online

19.5.5.1 Description

Gets the Admin Address value from the config.cfg file and displays it. The value can be of the form abc@xyz.def.

19.5.5.2 Syntax

`getUCMSysAdminAddress()`

or

`getUCMSysAdminAddress(`*`application_name`*`)`

19.5.5.3 Example

The following command displays the Admin Address value for the application "Content Server":

```
getUCMSysAdminAddress('Content Server')
mymail@mycompany.com
```

19.5.6 setUCMSysAdminAddress

Use with WLST: Online

19.5.6.1 Description

Sets the Admin Address value in the config.cfg file. The Admin Address can be of the form abc@xyz.def.

19.5.6.2 Syntax

`setUCMSysAdminAddress(`*`value`*`)`

or

`setUCMSysAdminAddress(`*`value,application_name`*`)`

19.5.6.3 Example

The following command sets the Admin Address value for the application "Content Server":

```
setUCMSysAdminAddress(mymail@mycompany.com,'Content Server')
```

19.6 Additional Commands

Use the commands in Table 19–4 to configure additional settings to monitor the Oracle WebCenter Content Server instance.

Table 19–4 WLST Additional Configuration Commands

Use this command...	To...	Use with WLST...
getUCMCSVersion	Display the version number.	Online
getUCMServerUptime	Display the uptime value.	Online

19.6.1 getUCMCSVersion

Use with WLST: Online

19.6.1.1 Description

Gets the version number of the Content Server running instance.

19.6.1.2 Syntax

`getUCMCSVersion()`

or

`getUCMCSVersion(`*`application_name`*`)`

19.6.1.3 Example

The following command displays the version number of the active instance of the application "Content Server":

```
getUCMCSVersion('Content Server')
11g R1
```

19.6.2 getUCMServerUptime

Use with WLST: Online

19.6.2.1 Description

Gets the amount of time the Content Server instance has been up.

19.6.2.2 Syntax

```
getUCMServerUptime()
```

or

```
getUCMServerUptime(application_name)
```

19.6.2.3 Example

The following command displays the amount of time the application "Content Server" has been up:

```
getUCMServerUptime('Content Server')
00H:01 Min:12 Sec:255 MilliSeconds
```

20

Enterprise Scheduling Service (ESS) Custom WLST Commands

Use the Oracle Enterprise Scheduling Service (ESS) commands in the categories listed in Table 20–1 to manage ESS configuration, servers, logs, and job requests.

> **Note:** To use these ESS custom WLST commands, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide*.

When running these WLST commands, you must have the following JARs on your classpath:

- *MW_HOME*/oracle_common/modules/oracle.jmx_11.1.1/jmxframework.jar
- *WL_HOME*/server/lib/weblogic.jar
- *MW_HOME*/*ORACLE_HOME*/ess/lib/ess-admin.jar

20.1 ESS Custom Commands

Use the ESS commands listed in Table 20–1 to manage the ESS server, configuration, job requests, and logs. In the **Use with WLST** column, "online" means the command can only be used when connected to a running administration server. "Offline" means the command can only be used when not connected to a running server.

Table 20–1 Oracle Enterprise Scheduling Service Management Commands

Use this command...	To...	Use with WLST...
essGetRequestContent	Get the log and output data files for a request after its execution is completed.	Online
essManageRequest	Cancel, recover, or complete request state manually.	Online
essManageRuntimeConfig	Add, modify, delete and display various configuration parameters.	Online
essManageServer	Start, stop or get status of the ESS application running on the server.	Online
essQueryRequests	Search and list requests based upon hosting application name, state or elapsed time of execution.	Online

20.1.1 essGetRequestContent

Command Category: ESS

Use with WLST: Online

20.1.1.1 Description

Get the log and output data files for a request after its execution is completed.

20.1.1.2 Syntax

```
essGetOutputContent(requestId, contentType, logLines, outDir)
```

Argument	Definition
requestId	The request ID.
contentType	Type of the content to handle. Can be LOG, OUTPUT, BINARY_OUTPUT or TEXT_OUTPUT. By default, the OUTPUT contentType checks for both BINARY_OUTPUT and TEXT_OUTPUT contents.
logLines	Optional. The number of lines to be read form the request log. Default is 1000.
outDir	Optional. The absolute path of the output directory to dump the output files into. Default is the current directory.

20.1.1.3 Examples

To get the request log for request ID 123.

```
essGetOutputContent(123, "LOG")
```

To get all the output of request 123.

```
essGetOutputContent(123, "OUTPUT")
```

To get all the output of request 123 and save it in directory /tmp.

```
essGetOutputContent(123, "OUTPUT", outDir="/tmp")
```

To get all the text output of request 123 and save it in directory /tmp.

```
essGetOutputContent(123, "TEXT_OUTPUT", outDir="/tmp")
```

To get all the binary output of request 123 and save it in directory /tmp.

```
essGetOutputContent(123, "BINARY_OUTPUT", outDir="/tmp")
```

To get first 100 lines of the request log for request id 123.

```
essGetOutputContent(123, "LOG", logLines=100)
```

20.1.2 essManageRequest

Command Category: ESS

Use with WLST: Online

20.1.2.1 Description

Cancel, recover, or complete request state manually.

20.1.2.2 Syntax

```
essManageRequest(requestId, operation, asyncStatus, statusMessage)
```

Argument	Definition
requestId	The request ID.
operation	The operation to perform: CANCEL, RECOVER, or COMPLETE.
asyncStatus	Mandatory when the COMPLETE operation is specified. The status to set for the given request. Must be one of the following: ■ BIZ_ERROR ■ CANCEL ■ ERROR ■ ERROR_MANUAL_RECOVERY ■ PAUSE ■ SUCCESS ■ UPDATE ■ WARNING
statusMessage	Optional. The qualifying status message to describe the operation.

20.1.2.3 Examples

To cancel request 123.

```
essManageRequest(123, "CANCEL")
```

To recover request 123.

```
essManageRequest(123, "RECOVER")
```

To complete request 123.

```
essManageRequest(123, "COMPLETE", asyncStatus="ERROR", statusMessage="Completed by
Admin")
```

20.1.3 essManageRuntimeConfig

Command Category: ESS

Use with WLST: Online

20.1.3.1 Description

Add, modify, delete and display various configuration parameters.

20.1.3.2 Syntax

```
essManageRuntimeConfig(app, type, operation, name, val)
```

Argument	Definition
app	The hosting application name for managing runtime configuration.
type	The type of configuration property. Can be either APP or ESS.
operation	Optional. The operation to perform. Value can be one of the following: add (add), mod (modify), del (delete), get (get), or getall (get all). The default is getall.

Argument	Definition
name	Optional when the getall value is used for operation. The name of the configuration parameter.
val	Optional when the del, get, or getall value is used for operation. The value to set for the parameter.

20.1.3.3 Examples

To add an ENV parameter "foo" with value "bar".

```
essManageRuntimeConfig("myapp", "APP", operation="add", name="foo", val="bar")
```

To get the value of the ENV parameter "foo".

```
essManageRuntimeConfig("myapp", "APP", operation="get", name="foo")
```

To get the list of all the ENV parameters.

```
essManageRuntimeConfig("myapp", "APP", operation="getall")
essManageRuntimeConfig("myapp", "APP")
```

To modify the value of the ENV parameter "foo" to "barone".

```
essManageRuntimeConfig("myapp", "APP", operation="mod", name="foo", val="barone")
```

To delete the ENV parameter "foo".

```
essManageRuntimeConfig("myapp", "APP", operation="del", name="foo")
```

To show all parameters of type ESS.

```
essManageRuntimeConfig("myapp", "ESS")
```

20.1.4 essManageServer

Command Category: ESS

Use with WLST: Online

20.1.4.1 Description

Start, stop or get status of the ESS application running on the server. Starting the ESS application means to start the ESS processor thread so that request processing can start. Stopping ESS means to stop or quiesce the ESS processor so that no new requests are processed.

If connected to the WLS Administration Server in a cluster, this command would operate upon all nodes in the cluster.

20.1.4.2 Syntax

```
essManageServer(operation)
```

Argument	Definition
operation	The operation to perform. One of START, STOP, or STATUS.

20.1.4.3 Examples

To stop ESS.

```
essManageServer("STOP")
```

To get the current state of the ESS processor.

```
essManageServer("STATUS")
```

20.1.5 essQueryRequests

Command Category: ESS

Use with WLST: Online

20.1.5.1 Description

Search and list requests based on hosting application name, state or elapsed time of execution. This command can be used to find long running requests.

20.1.5.2 Syntax

```
essQueryRequests(app, state, days, hours, minutes)
```

Argument	Definition
app	Optional. The name of the hosting application.
state	Optional. The request state. Can be one of the following (default is RUNNING): ■ BLOCKED ■ CANCELLED ■ CANCELLING ■ COMPLETED ■ ERROR ■ ERROR_AUTO_RETRY ■ ERROR_MANUAL_RECOVERY ■ EXPIRED ■ FINISHED ■ HOLD ■ PAUSED ■ PENDING_VALIDATION ■ READY ■ RUNNING ■ SCHEDULE_ENDED ■ SUCCEEDED ■ VALIDATION_FAILED ■ WAIT ■ WARNING
days	Optional. Specifies the time in days.
hours	Optional. Specifies the time in hours.
minutes	Optional. Specifies the time in minutes.

20.1.5.3 Examples

To get all the requests in RUNNING state.

```
essQueryRequests()
essQueryRequests(state="RUNNING")
```

To get all CANCELLED requests.

```
essQueryRequests(state="CANCELLED")
```

To get all requests running for more than 2 days.

```
essQueryRequests(days=2)
```

To get all requests running for more than 10 hours.

```
essQueryRequests(hours=10)
```

To get all requests running for the application "myapp".

```
essQueryRequests(appName="myapp")
```