Oracle® Fusion Middleware

Infrastructure Security WLST Command Reference

12*c* (12.1.3)

E47836-02

December 2014

This document describes the Oracle Fusion Middleware Infrastructure Security commands available to use with the WebLogic Scripting Tool (WLST).

ORACLE

Oracle Fusion Middleware Infrastructure Security WLST Command Reference, 12*c* (12.1.3)

E47836-02

Primary Author: Carlos Subi

Contributing Authors: Vinaye Misra, Devanshi Mohan

Contents

Preface v

Audience v
Documentation Accessibility v
Related Documentation v
Conventions v

1 Introduction and Roadmap

1.1 Document Scope and Audience 1-1
1.2 Guide to This Document 1-1

2 Infrastructure Security Custom WLST Commands

2.1 Infrastructure Commands 2-1
2.1.1 OPSS Security Store Commands 2-1
2.1.2 Audit Configuration Commands 2-22
2.1.3 OPSS Keystore Service Commands 2-31
2.1.4 Identity Directory Service Commands 2-42
2.1.5 Library Oracle Virtual Directory (libOVD) Commands 2-56

3 SSL Configuration WLST Commands

3.1 About SSL Configuration Commands 3-1
3.2 Properties Files for SSL 3-2
3.2.1 Structure of Properties Files 3-2
3.2.2 Examples of Properties Files 3-3
3.3 configureSSL 3-4
3.3.1 Description 3-4
3.3.2 Syntax 3-4
3.3.3 Examples 3-5
3.4 getSSL 3-5
3.4.1 Description 3-5
3.4.2 Syntax 3-5
3.4.3 Example 3-5

4 Wallet Configuration WLST Commands

4.1 Background for WLST Wallet Commands 4-1

4.2 WLST Reference for Wallet Management 4-2
4.2.1 addCertificateRequest 4-2
4.2.2 addSelfSignedCertificate 4-3
4.2.3 changeWalletPassword 4-3
4.2.4 createWallet 4-4
4.2.5 deleteWallet 4-4
4.2.6 exportWallet 4-5
4.2.7 exportWalletObject 4-6
4.2.8 getWalletObject 4-7
4.2.9 importWallet 4-8
4.2.10 importWalletObject 4-8
4.2.11 listWalletObjects 4-9
4.2.12 listWallets 4-10
4.2.13 removeWalletObject 4-10

Preface

This guide describes the security WebLogic Scripting Tool (WLST) commands for Oracle Platform Security Services (OPSS).

Audience

The intended audience of this guide are experienced Java developers, administrators, deployers, and application managers who want to use the security OPSS commands.

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.

Related Documentation

Additional information is found in the following documents:

- *Securing Applications with Oracle Platform Security Services*
- *Administering Oracle Fusion Middleware*
- *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*

For a comprehensive list of Oracle documentation or to search for a particular topic within Oracle documentation libraries, see http://www.oracle.com/technology/documentation/index.html.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action.

Convention	Meaning
italic	Italic type indicates book titles, emphasis, terms defined in text, or placeholder variables for which you supply particular values.
`monospace`	Monospace type within a paragraph indicates commands, URLs, Java class names and method names, file and directory names, text that appears on the screen, or text that you enter.

1 Introduction and Roadmap

This chapter describes the audience for and contents and organization of this guide—*Infrastructure Security WLST Command Reference*.

This chapter includes the following sections:

- Document Scope and Audience
- Guide to This Document

1.1 Document Scope and Audience

This document describes all of the Infrastructure Security custom WLST commands that are available to use with the WebLogic Scripting Tool (WLST).

> **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 and Fusion Middleware products are 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, "Infrastructure Security Custom WLST Commands," provides detailed descriptions for each of the custom WLST commands for audit configuration, SSL configuration, Oracle Identify Federation, Directory Integration Platform, OPSS, Oracle Keystore Service, Identity Directory Service, and Library Oracle Virtual Directory (libOVD).

2

Infrastructure Security Custom WLST Commands

This chapter describes the Oracle Fusion Middleware Infrastructure Security WLST commands.

It contains the following section:

- Infrastructure Commands

For additional information about Oracle Platform Security Services (OPSS), see *Securing Applications with Oracle Platform Security Services*.

> **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 *Administering Oracle Fusion Middleware*.

2.1 Infrastructure Commands

The infrastructure WLST security commands are divided into the following categories:

Table 2–1 WLST Command Categories

Command Category	Description
OPSS Security Store Commands	Manage domain and credential domain stores and migrate domain policy store.
Audit Configuration Commands	View and manage audit policies and the audit repository configuration
OPSS Keystore Service Commands	Manage the OPSS keystore service.
Identity Directory Service Commands	Manage Identity Directory Service entity attributes, entity definitions, relationships, and default operational configurations.
Library Oracle Virtual Directory (libOVD) Commands	View and manage Library Oracle Virtual Directory (libOVD) configurations associated with a particular OPSS context.

2.1.1 OPSS Security Store Commands

Use the WLST security commands listed in Table 2–2 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 2–2 WLST Security Commands

Use this command...	To...	Use with WLST...
addBootStrapCredential	Add a credential to the bootstrap credential store	Offline
addResourceToEntitlement	Add a resource to an entitlement.	Online
createAppRole	Create a new application role.	Online
createCred	Create a new credential.	Online
createEntitlement	Create an entitlement.	Online
createResource	Create a resource.	Online
createResourceType	Create a new resource type.	Online
deleteAppPolicies	Remove all policies in an application.	Online
deleteAppRole	Remove an application role.	Online
deleteCred	Remove a credential.	Online
deleteEntitlement	Remove an entitlement.	Online
deleteResource	Remove a resource.	Online
deleteResourceType	Remove an existing resource type.	Online
exportEncryptionKey	Export the domain encryption key to the file `ewallet.p12`.	Offline
getEntitlement	List an entitlement.	Online
getResourceType	Fetch an existing resource type.	Online
grantAppRole	Add a principal to a role.	Online
grantEntitlement	Create an entitlement.	Online
grantPermission	Create a new permission.	Online
importEncryptionKey	Import the encryption key in file `ewallet.p12` to the domain.	Offline
listAppRoles	List all roles in an application.	Online
listAppRolesMembers	List all members in an application role.	Online
listAppStripes	List application stripes in policy store.	Online
listCodeSourcePermissions	List permissions assigned to a source code in global policies.	Online
listEntitlement	List an entitlement.	Online
listEntitlements	List entitlements in an application stripe.	Online
listPermissions	List all permissions granted to a principal.	Online
listResourceActions	List actions in a resource.	Online
listResourceTypes	List resource types in an application stripe.	Online
listResources	List resources in an application stripe.	Online
listSecurityStoreInfo	List the type and location of the OPSS security store, and the user allowed to access it.	Offline
migrateSecurityStore	Migrate policies or credentials from a source repository to a target repository.	Offline
modifyBootStrapCredential	Update bootstrap credential store	Offline

Table 2–2 (Cont.) WLST Security Commands

Use this command...	To...	Use with WLST...
reassociateSecurityStore	Reassociate policies and credentials to an LDAP repository	Online
restoreEncryptionKey	Restore the domain encryption key as it was before the last importing.	Offline
revokeAppRole	Remove a principal from a role.	Online
revokeEntitlement	Remove an entitlement.	Online
revokePermission	Remove a permission.	Online
revokeResourceFromEntitlement	Remove a resource from an entitlement	Online
rollOverEncryptionKey	Replace the current domain encryption key with a new one.	Offline
updateCred	Modify the attribute values of a credential.	Online
updateTrustServiceConfig	Update the configuration of the trust service.	Online

2.1.1.1 addBootStrapCredential

Offline command that adds a credential to the bootstrap credential store.

2.1.1.1.1 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`.

2.1.1.1.2 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.

2.1.1.1.3 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')
```

2.1.1.2 addResourceToEntitlement

Online command that adds a resource with specified actions to an entitlement.

2.1.1.2.1 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.

2.1.1.2.2 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.

2.1.1.2.3 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")
```

2.1.1.3 createAppRole

Online command that creates a new application role.

2.1.1.3.1 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`.

2.1.1.3.2 Syntax `createAppRole(appStripe, appRoleName)`

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

2.1.1.3.3 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")
```

2.1.1.4 createCred

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

2.1.1.4.1 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.

2.1.1.4.2 Syntax Optional arguments are enclosed in square brackets.

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

Argument	Definition
map	Specifies a map name (folder).

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

2.1.1.4.3 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")
```

2.1.1.5 createEntitlement

Online command that creates a new entitlement.

2.1.1.5.1 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.

2.1.1.5.2 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.

2.1.1.5.3 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")
```

2.1.1.6 createResource

Online command that creates a new resource.

2.1.1.6.1 Description Creates a resource of a specified type in a specified application stripe. The passed resource type must exist in the passed application stripe.

2.1.1.6.2 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.

2.1.1.6.3 Example The following invocation creates the resource myResource in the stripe myApplication:

```
wls:/mydomain/serverConfig> createResource(appStripe="myApplication",
name="myResource", type="myResType", displayName="myNewResource")
```

2.1.1.7 createResourceType

Online command that creates a new resource type in the domain policy store within a given application stripe.

2.1.1.7.1 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`.

2.1.1.7.2 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 ','.

2.1.1.7.3 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=";"])
```

2.1.1.8 deleteAppPolicies

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

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

2.1.1.8.2 Syntax `deleteAppPolicies(appStripe)`

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

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

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

2.1.1.9 deleteAppRole

Online command that removes an application role.

2.1.1.9.1 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`.

2.1.1.9.2 Syntax `createAppRole(appStripe, appRoleName)`

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

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

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

2.1.1.10 deleteEntitlement

Online command that deletes an entitlement.

2.1.1.10.1 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.

2.1.1.10.2 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.

2.1.1.10.3 Example The following invocation deletes the entitlement myEntitlement in the stripe myApplication:

```
wls:/mydomain/serverConfig> deleteEntitlement(appStripe="myApplication",
```

```
name="myEntitlement")
```

2.1.1.11 deleteCred

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

2.1.1.11.1 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`.

2.1.1.11.2 Syntax `deleteCred(map,key)`

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

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

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

2.1.1.12 deleteResource

Online command that deletes a resource.

2.1.1.12.1 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.

2.1.1.12.2 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.

2.1.1.12.3 Example The following invocation deletes the resource myResource in the stripe myApplication:

```
wls:/mydomain/serverConfig> deleteResource(appStripe="myApplication",
name="myResource", type="myResType")
```

2.1.1.13 deleteResourceType

Online command that removes a resource type from the domain policy store within a given application stripe.

2.1.1.13.1 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`.

2.1.1.13.2 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.

2.1.1.13.3 Example The following invocation removes the resource type myResType from the stripe myApplication:

```
wls:/mydomain/serverConfig> deleteResourceType(appStripe="myApplication",
resourceTypeName="myResType")
```

2.1.1.14 exportEncryptionKey

Offline command that extracts the encryption key from a domain's bootstrap wallet to the file `ewallet.p12`.

2.1.1.14.1 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`.

2.1.1.14.2 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.

2.1.1.14.3 Example The following invocation writes the file `ewallet.p12` in the directory myDir:

```
exportEncryptionKey(jpsConfigFile="pathName", keyFilePath="myDir"
,keyFilePassword="password")
```

2.1.1.15 getEntitlement

Online command that gets an entitlement.

2.1.1.15.1 Description Returns the name, display name, and all the resources (with their actions) of an entitlement in an application stripe.

2.1.1.15.2 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.

2.1.1.15.3 Example The following invocation returns the information of the entitlement myEntitlement in the stripe myApplication:

```
wls:/mydomain/serverConfig> getEntitlement(appStripe="myApplication",
name="myEntitlement")
```

2.1.1.16 getResourceType

Online command that fetches a resource type from the domain policy store within a given application stripe.

2.1.1.16.1 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`.

2.1.1.16.2 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.

2.1.1.16.3 Example The following invocation fetches the resource type myResType from the stripe myApplication:

```
wls:/mydomain/serverConfig> getResourceType(appStripe="myApplication",
resourceTypeName="myResType")
```

2.1.1.17 grantAppRole

Online command that adds a principal to a role.

2.1.1.17.1 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`.

2.1.1.17.2 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.

2.1.1.17.3 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")
```

2.1.1.18 grantEntitlement

Online command that creates a new entitlement.

2.1.1.18.1 Description Creates a new entitlement with a specified principal in a specified application stripe.

2.1.1.18.2 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.

2.1.1.18.3 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")
```

2.1.1.19 grantPermission

Online command that creates a new permission.

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

2.1.1.19.2 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.

2.1.1.19.3 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")
```

2.1.1.20 importEncryptionKey

Offline command that imports keys from the specified ewallet.p12 file into the domain.

2.1.1.20.1 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`.

2.1.1.20.2 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.

2.1.1.20.3 Example `importEncryptionKey(jpsConfigFile="`*pathName*`", keyFilePath="`*dirloc*`" ,keyFilePassword="`*password*`")`

2.1.1.21 listAppRoles

Online command that lists all roles in an application.

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

2.1.1.21.2 Syntax `listAppRoles(appStripe)`

Argument	Definition
appStripe	Specifies an application stripe.

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

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

2.1.1.22 listAppRolesMembers

Online command that lists all members in a role.

2.1.1.22.1 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`.

2.1.1.22.2 Syntax `listAppRoleMembers(appStripe, appRoleName)`

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

2.1.1.22.3 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")
```

2.1.1.23 listAppStripes

Online or offline command that lists the application stripes in the policy store.

2.1.1.23.1 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.

2.1.1.23.2 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 *.

2.1.1.23.3 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*")
```

2.1.1.24 listCodeSourcePermissions

Online command that lists permissions assigned to a source code in global policies.

2.1.1.24.1 Description This command allows listing codebase permissions in global policies.

2.1.1.24.2 Syntax `listCodeSourcePermissions([codeBase="codeUrl"])`

Argument	Definition
codeBaseURL	Specifies the name of the grantee codebase URL.

2.1.1.24.3 Examples The following invocation returns the list permissions assigned to a code source in all global policies:

```
wls:/mydomain/serverConfig>
listCodeSourcePermissions(codeBaseURL="file:/tmp/lib/myJars.jar")
```

2.1.1.25 listEntitlement

Online command that lists an entitlement in a specified application stripe.

2.1.1.25.1 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.

2.1.1.25.2 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.

2.1.1.25.3 Example The following invocation lists all entitlements in the stripe myApplication:

```
wls:/mydomain/serverConfig> listEntitlement(appStripe="myApplication")
```

2.1.1.26 listEntitlements

Online command that lists the entitlements in an application stripe.

2.1.1.26.1 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.

2.1.1.26.2 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.

2.1.1.26.3 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")
```

2.1.1.27 listPermissions

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

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

2.1.1.27.2 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.

2.1.1.27.3 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")
```

2.1.1.28 listResourceActions

Online command that lists the resources and actions in an entitlement.

2.1.1.28.1 Description Lists the resources and actions in an entitlement within an application stripe.

2.1.1.28.2 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.

2.1.1.28.3 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")
```

2.1.1.29 listResources

Online command that lists resources in a specified application stripe.

2.1.1.29.1 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.

2.1.1.29.2 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.

2.1.1.29.3 Example The following invocation lists all resources of type myResType in the stripe myApplication:

```
wls:/mydomain/serverConfig> listResources(appStripe="myApplication",
type="myResType")
```

2.1.1.30 listResourceTypes

Online command that lists resource types.

2.1.1.30.1 Description Lists all the resource types in a specified application stripe.

2.1.1.30.2 Syntax `listResourceTypes(appStripe="appStripeName")`

Argument	Definition
appStripe	Specifies the application stripe where the resource types are located.

2.1.1.30.3 Example The following invocation lists all resource types in the stripe myApplication:

```
wls:/mydomain/serverConfig> listResourceTypes(appStripe="myApplication")
```

2.1.1.31 listSecurityStoreInfo

Offline command that lists the type, the location, and the administrative user of the domain security store.

2.1.1.31.1 Description The script runs in offline mode and outputs the type of the OPSS security store (file, OID, or DB), its location, and the user allowed to access it (typically a security administrator).

2.1.1.31.2 Syntax `listSecurityStoreInfo(domainConfig="configFilePath")`

Argument	Definition
domainConfig	Specifies the full absolute path to the OPSS configuration file jps-config.xml; the file jps-config-jse.xml is also expected to be in the passed directory.

2.1.1.31.3 Example The following invocation returns the type, location, and administrative user of the OPSS policy store:

```
wls:/mydomain/serverConfig>
listSecurityStoreInfo(domainConfig="/home/myConfigPathDirectory/config/fmwconfig")
```

The following lines illustrate a sample output generated by this command:

```
For jps-config.xml
Store Type: DB_ORACLE
Location/Endpoint: jdbc:oracle:thin:@adc2120515.us.myComp.com:1555/OWSM.US.COM
User: DEV_OPSS
Datasource: jdbc/OpssDataSource
For jps-config-jse.xml
Store Type: DB_ORACLE
Location/Endpoint: jdbc:oracle:thin:@adc2120515.us.myComp.com:1521/OWSM.US.COM
User: DEV_OPSS
```

2.1.1.32 migrateSecurityStore

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

2.1.1.32.1 Description Migrates security artifacts from a source repository to a target repository. For full details, see Migrating with the Script migrateSecurityStore.

2.1.1.33 modifyBootStrapCredential

Offline command that updates a bootstrap credential store.

2.1.1.33.1 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.

2.1.1.33.2 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.

2.1.1.33.3 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.

2.1.1.34 reassociateSecurityStore

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

2.1.1.34.1 Description The script `reassociateSecurityStore` migrates the OPSS security store from a source to a target LDAP- or DB-based store, and it resets services in the files `jps-config.xml` and `jps-config-jse.xml` to the target repository. It also allows specifying that the OPSS security store be shared with that in a different domain (see optional argument `join` below). The OPSS binaries and the target policy store must have compatible versions.

For complete details and samples see *Securing Applications with Oracle Platform Security Services*.

2.1.1.35 restoreEncryptionKey

Offline command to restore the domain credential encryption key.

2.1.1.35.1 Description Restores the state of the domain bootstrap keys as it was before running importEncryptionKey.

2.1.1.35.2 Syntax `restoreEncryptionKey(jpsConfigFile)`

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run.

2.1.1.35.3 Example `restoreEncryptionKey(jpsConfigFile="`*pathName*`")`

2.1.1.36 revokeAppRole

Online command that removes a principal from a role.

2.1.1.36.1 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`.

2.1.1.36.2 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.

2.1.1.36.3 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")
```

2.1.1.37 revokeEntitlement

Online command that deletes an entitlement.

2.1.1.37.1 Description Deletes an entitlement and revokes the entitlement from the principal in a specified application stripe.

2.1.1.37.2 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.

2.1.1.37.3 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")
```

2.1.1.38 revokePermission

Online command that removes a permission.

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

2.1.1.38.2 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.
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.

Argument	Definition
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.

2.1.1.38.3 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")
```

2.1.1.39 revokeResourceFromEntitlement

Online command that removes a resource from an entitlement.

2.1.1.39.1 Description Removes a resource from an entitlement in a specified application stripe.

2.1.1.39.2 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.

2.1.1.39.3 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")
```

2.1.1.40 rollOverEncryptionKey

Offline command that changes the domain encryption key.

2.1.1.40.1 Description This offline script replaces the current domain OPSS encryption key with a new one; the current key is not deleted but archived, since it is used to decrypt data that was encrypted using that key.

Note the following important points:

- This command should be executed from the administration server in the domain. No server restart is needed after its execution.

- If the domain is the only domain accessing the security store, nothing else is required.
- However, if two or more domains share the security store, the newly generated key should be exported from the domain where the script was run and imported into each of the other domains sharing the security store, using the scripts exportEncryptionKey and importEncryptionKey.

2.1.1.40.2 Syntax `rollOVerEncryptionKey(jpsConfigFile="pathName")`

Argument	Definition
jpsConfigFile	Specifies the location of the file jps-config.xml; either relative to the location where the script is run, or the full path.

2.1.1.40.3 Example The following invocation lists all resource types in the stripe myApplication:

```
wls:/mydomain/serverConfig> rollOverEncryptionKey(jpsConfigFile="myConfig")
```

2.1.1.41 updateCred

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

2.1.1.41.1 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.

2.1.1.41.2 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.

2.1.1.41.3 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")
```

2.1.1.42 updateTrustServiceConfig

Online command that updates the configuration of the domain trust service service with the values passed in a property file.

2.1.1.42.1 Description Updates the trust service domain configuration. In the event of an error, the command returns a `WLSTException`.

2.1.1.42.2 Syntax `updateTrustServiceConfig([providerName="<the provider name>",]`

```
                                    propsFile="<path of properties file>")
```

Argument	Definition
providerName	Specifies the name of the trust service provider; optional; if unspecified, it defaults to `trust.provider.embedded`.
propsFile	Specifies the path to the file where the property values are set.

Here is a sample property file:

```
trust.keystoreType=KSS
trust.keyStoreName=kss://<stripeName>/<keystoreName>
trust.trustStoreName=kss://<stripeName>/<truststoreName>
trust.aliasName=<aliasName>
trust.issuerName=<aliasName>
```

Note that the list of specified properties differs according to the value of the property `trust.keystoreType`. The type can be `KSS` or `JKS`; if a property is set to the empty string, then that property is removed from the trust service configuration. For the list of available properties, see section Trust Service Properties.

2.1.1.42.3 Example The following invocation updates the trust store service with the specifications in the file myProps:

```
wls:/mydomain/serverConfig> updateTrustServiceConfig(providerName="myProvider",
propsFile="myProps")
```

2.1.2 Audit Configuration Commands

Use the WLST commands listed in Table 2–3 to view and manage audit policies and the audit repository configuration.

Table 2–3 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
createAuditDBView	Create an audit definitions view in the database.	Online
listAuditComponents	List components that can be audited.	Online
registerAudit	Registers audit definitions for a specified component in the audit store.	Online
deregisterAudit	Removes audit definitions of a specified component from the audit store.	Online

For more information, see the *Securing Applications with Oracle Platform Security Services*.

2.1.2.1 getNonJavaEEAuditMBeanName

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

2.1.2.1.1 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.

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

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

2.1.2.1.3 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')
```

2.1.2.2 getAuditPolicy

Online command that displays the audit policy settings.

2.1.2.2.1 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 HTTP Server.

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

2.1.2.2.2 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.

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

```
wls:/mydomain/serverConfig> getAuditPolicy(componentType='JPS');
```

```
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
```

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

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

2.1.2.3 setAuditPolicy

Online command that updates an audit policy.

2.1.2.3.1 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 HTTP Server.

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

2.1.2.3.2 Syntax

```
setAuditPolicy([mbeanName],[filterPreset],[addSpecialUsers],
[removeSpecialUsers],[addCustomEvents],[removeCustomEvents], [componentType],
[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. The audit runtime policy for the component is registered in the audit store. If not specified, the audit configuration defined in jps-config.xml is modified.
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".

2.1.2.3.3 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',componentType='JPS
')

wls:/mydomain/serverConfig> getAuditPolicy(componentType='JPS');
Already in Domain Runtime Tree

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

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',componentType='JPS);
Already in Domain Runtime Tree
Audit Policy Information updated successfully

wls:/IDMDomain/domainRuntime> getAuditPolicy(componentType='JPS')
Already in Domain Runtime Tree
FilterPreset:Low
Max Log File Size:104857600
```

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

```
wls:/IDMDomain/domainRuntime>setAuditPolicy(filterPreset='Custom',
componentType='JPS',addCustomEvents='Authorization:CheckPermission,
CheckSubject;CredentialManagement:CreateCredential,DeleteCredential');
Already in Domain Runtime Tree

Audit Policy Information updated successfully
wls:/IDMDomain/domainRuntime> getAuditPolicy(componentType='JPS');
Already in Domain Runtime Tree

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

2.1.2.4 getAuditRepository

Online command that displays audit repository settings.

2.1.2.4.1 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.

2.1.2.4.2 Syntax `getAuditRepository`

2.1.2.4.3 Example The following command displays audit repository configuration:

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

Repository Type:File
```

2.1.2.5 setAuditRepository

Online command that updates audit repository settings.

2.1.2.5.1 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`).

2.1.2.5.2 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.

2.1.2.5.3 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='jdbc/AuditDB',interval='14')
```

2.1.2.6 listAuditEvents

Online command that displays a component's audit events.

2.1.2.6.1 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.

2.1.2.6.2 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.

2.1.2.6.3 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
```

...

2.1.2.7 exportAuditConfig

Online command that exports a component's audit configuration.

2.1.2.7.1 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.

2.1.2.7.2 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.

2.1.2.7.3 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')
```

2.1.2.8 importAuditConfig

Online command that imports a component's audit configuration.

2.1.2.8.1 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.

2.1.2.8.2 Syntax `importAuditConfig([mbeanName],fileName, [componentType])`

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

Argument	Definition
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.

2.1.2.8.3 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')
```

2.1.2.9 createAuditDBView

Creates a SQL script that can generate a view for audit in the database.

2.1.2.9.1 Description This command generates a SQL script that you can use to create a database view of the audit definitions of a specified component. The script is written to the specified file and also printed out to the console.

Upon execution, the result of the SQL script depends on the audit model at your site:

- If using the 11.1.1.6.0 model, and the component is registered in the audit store, the script creates a view using the system component tables (IAU_COMMON, IAU_USERSESSION, IAU_AUDITSERVICE and IAU_CUSTOM) for the specified component.
- If using the pre-11.1.1.6.0 model, the component is not registered in the audit store but its event definitions reside in the component_events.xml file (in the oracle_common/modules/oracle.iau_11.1.1/components/*componentType* dir), and the view is created using the IAU_BASE and component tables.

2.1.2.9.2 Syntax `createAuditDBView(fileName, componentType)`

Argument	Definition
fileName	Specifies the path and file name to which the SQL script is written.
componentType	The component whose definitions are the basis of the view.

2.1.2.9.3 Example

```
wls:/mydomain/serverConfig>
createAuditDBView(fileName="/tmp/JPSAuditView.sql", componentType="JPS")
```

2.1.2.10 listAuditComponents

Lists components that can be audited.

2.1.2.10.1 Description This command creates a list of the components that can be audited. It lists components registered in the audit store using both the 11.1.1.6.0 model and the pre-11.1.1.6.0 model.

2.1.2.10.2 Syntax `listAuditComponents(fileName)`

Argument	Definition
fileName	Specifies the path and file name to which the output is written.

2.1.2.10.3 Example `listAuditComponents(fileName = "/tmp/complist.txt")`

2.1.2.11 registerAudit

Registers the specified component in the audit store.

2.1.2.11.1 Description Adds the event definition and translation content for a specified component to the audit store. If you try to register using the pre-11.1.1.6.0 audit XML schema definition, it is upgraded to the 11.1.1.6.0 XML schema definition and then registered with the audit store.

2.1.2.11.2 Syntax `registerAudit(xmlFile, [xlfFile], componentType, [mode=OVERWRITE|UPGRADE])`

Argument	Definition
xmlFile	Specifies the Component Event definition file.
xlfFile	Specifies the component xlf jar file. Optional.
componentType	Specifies the component to be registered.
mode	OVERWRITE or UPGRADE. Default is UPGRADE.

2.1.2.11.3 Example

```
wls:/mydomain/serverConfig>registerAudit(xmlFile="/tmp/comp.xml",
xmlFile="/tmp/comp_xlf.jar", componentType="AuditApp", mode="UPGRADE")
```

2.1.2.12 deregisterAudit

Removes the event definition and translation content for the specified component from the audit store.

2.1.2.12.1 Description Removes an existing event definition and translation content for a specified component or application from the audit store.

2.1.2.12.2 Syntax `deregisterAudit(componentType)`

Argument	Definition
componentType	Specifies the component whose definitions are to be removed.

2.1.2.12.3 Example `deregisterAudit(componentType="AuditApp")`

2.1.3 OPSS Keystore Service Commands

This section contains commands used with the OPSS keystore service.

> **Note:** You need to acquire an OPSS handle to use keystore service commands; this handle is denoted by 'svc' in the discussion that follows. For details, see Managing Keys and Certificates with the Keystore Service in *Securing Applications with Oracle Platform Security Services*.

Table 2–4 lists the WLST commands used to manage the keystore service.

Table 2–4 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.
listKeyStoreAliases	List aliases in a keystore.
listKeyStores	List all the keystores in a stripe.
syncKeyStores	Synchronizes the keystores in the administration server with keystores in the security store.

2.1.3.1 changeKeyPassword

Changes a key password.

2.1.3.1.1 Description Changes the password for a key.

2.1.3.1.2 Syntax

```
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

2.1.3.1.3 Example This example changes the password on the key entry `orakey`:

```
changeKeyPassword(appStripe='system', name='keystore', password='password',
alias='orakey', currentkeypassword='currentkeypassword',
newkeypassword='newkeypassword')
```

2.1.3.2 changeKeyStorePassword

Changes the password of a keystore.

2.1.3.2.1 Description Changes the password of the specified keystore.

2.1.3.2.2 Syntax `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

2.1.3.2.3 Example This example changes the password for `keystore2`.

```
changeKeyStorePassword(appStripe='system', name='keystore2',
currentpassword='currentpassword', newpassword='newpassword')
```

2.1.3.3 createKeyStore

This keystore service command creates a new keystore.

2.1.3.3.1 Description Creates a new keystore on the given application stripe.

2.1.3.3.2 Syntax `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.

2.1.3.3.3 Example This example creates a keystore named `keystore1`.

```
createKeyStore(appStripe='system', name='keystore1', password='password',
permission=true)
```

2.1.3.4 deleteKeyStore

Deletes the named keystore.

2.1.3.4.1 Description This keystore service command deletes a specified keystore.

2.1.3.4.2 Syntax `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.

2.1.3.4.3 Example This example deletes the keystore named `keystore1`.

```
deleteKeyStore(appStripe='system', name='keystore1', password='password')
```

2.1.3.5 deleteKeyStoreEntry

Deletes a keystore entry.

2.1.3.5.1 Description This command deletes the specified entry in a keystore.

2.1.3.5.2 Syntax `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.

Argument	Definition
alias	Specifies the alias of the entry to be deleted
keypassword	Specifies the key password of the entry to be deleted

2.1.3.5.3 Example This example deletes a keystore entry denoted by alias `orakey`.

```
deleteKeyStoreEntry(appStripe='system', name='keystore2', password='password',
alias='orakey', keypassword='keypassword')
```

2.1.3.6 exportKeyStore

Exports a keystore to a file.

2.1.3.6.1 Description

Exports a keystore to the specified file.

2.1.3.6.2 Syntax

```
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. The value also applies to the output file, based on the current usage of the command: ■ For password-protected keystores of all types, this will be the password of the output file; ■ For permission-protected keystores of type JKS or JCEKS, this will be the password of the output file; ■ For permission-protected keystores of type OracleWallet, if the password value is non-empty, this will be the password of the output file; an empty value will create an auto-login wallet.
aliases	Comma separated list of aliases to be exported.
keypasswords	Specifies the password(s) of the key(s) being exported. The usage depends on the keystore type: ■ If type is JKS or JCEKS, and the keystore is permission-protected, this is a comma separated list of the key passwords corresponding to aliases in the output file. ■ If type is JKS or JCEKS, and the keystore is password-protected, this is a comma separated list of the key passwords corresponding to aliases in both the source keystore and the output file. ■ If type is OracleWallet, this parameter is ignored.
type	Exported keystore type. Valid values are 'JKS' or 'JCEKS' or 'OracleWallet'.
filepath	For type JKS or JCEKS, the absolute path of the file where the keystore is exported, including filename. For type OracleWallet, the absolute path of the directory where the keystore is exported.

2.1.3.6.3 Example

This example exports two aliases from the specified keystore.

```
exportKeyStore(appStripe='system', name='keystore2',
password='password',aliases='orakey,seckey',
keypasswords='keypassword1,keypassword2',
type='JKS',filepath='/tmp/file.jks')
```

This example exports a keystore to create an Oracle Wallet file:

```
exportKeyStore(appStripe='system', name='keystore2',
password='mypassword',aliases='orakey,seckey',
keypasswords='', type='OracleWallet',filepath='/tmp')
```

2.1.3.7 exportKeyStoreCertificate

Exports a certificate.

2.1.3.7.1 Description Exports a certificate, trusted certificate or certificate chain.

2.1.3.7.2 Syntax `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'.
filepath	Specifies the absolute path of the file where certificate, trusted certificate or certificate chain is exported.

2.1.3.7.3 Example This example exports a certificate corresponding to the `orakey` alias:

```
exportKeyStoreCertificate(appStripe='system', name='keystore2',
password='password', alias='orakey', keypassword='keypassword',
type='Certificate', filepath='/tmp/cert.txt')
```

2.1.3.8 exportKeyStoreCertificateRequest

Exports a certificate request.

2.1.3.8.1 Description Generates and exports a certificate request from a keystore.

2.1.3.8.2 Syntax `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.

2.1.3.8.3 Example This example exports a certificate request corresponding to the `orakey` alias.

```
exportKeyStoreCertificateRequest(appStripe='system', name='keystore2',
password='password', alias='orakey', keypassword='keypassword',
filepath='/tmp/certreq.txt')
```

2.1.3.9 generateKeyPair

Generates a key pair in a keystore.

2.1.3.9.1 Description Generates a key pair in a keystore and wraps it in a demo CA-signed certificate.

2.1.3.9.2 Syntax

```
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.

2.1.3.9.3 Example This example generates a keypair in `keystore2`.

```
generateKeyPair(appStripe='system', name='keystore2', password='password',
dn='cn=www.oracle.com', keysize='1024', alias='orakey', keypassword='keypassword')
```

2.1.3.10 generateSecretKey

Generates a secret key.

2.1.3.10.1 Description Generates a symmetric key in a keystore.

2.1.3.10.2 Syntax `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.

2.1.3.10.3 Example This example generates a keypair with keysize 128 in `keystore2`.

```
generateSecretKey(appStripe='system', name='keystore2', password='password',
algorithm='AES', keysize='128', alias='seckey', keypassword='keypassword')
```

2.1.3.11 getKeyStoreCertificates

Gets a certificate from the keystore.

2.1.3.11.1 Description Retrieves information about a certificate or trusted certificate.

2.1.3.11.2 Syntax `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.

2.1.3.11.3 Example This example gets certificates associated with `keystore3`.

```
getKeyStoreCertificates(appStripe='system', name='keystore3', password='password',
alias='orakey', keypassword='keypassword')
```

2.1.3.12 getKeyStoreSecretKeyProperties

Retrieves secret key properties.

2.1.3.12.1 Description Retrieves secret key properties like the algorithm.

2.1.3.12.2 Syntax
```
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.
keypassword	Specifies the secret key password.

2.1.3.12.3 Example This example gets properties for secret key `seckey`:

```
getKeyStoreSecretKeyProperties(appStripe='system', name='keystore3',
password='password', alias='seckey', keypassword='keypassword')
```

2.1.3.13 importKeyStore

Imports a keystore from file.

2.1.3.13.1 Description Imports a keystore from a system file.

2.1.3.13.2 Syntax
```
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 will reside.
name	Specifies the name of the keystore.
password	Specifies the keystore password. These rules apply: ■ If importing an auto-login Oracle Wallet file, no password is needed. ■ If importing a password-protected Oracle Wallet file (ewallet.p12), enter a password of minimum eight characters.
aliases	Specifies the comma-separated aliases of the entries to be imported from the file.
keypasswords	Specifies the passwords of the keys in the file. These rules apply: ■ If type is JKS or JCEKS, enter comma-separated passwords of the keys. ■ If type is OracleWallet, no password is needed. The key passwords will be the same as the keystore password.
type	Specifies the imported keystore type. Valid values are 'JKS' or 'JCEKS' or 'OracleWallet'.

Argument	Definition
filepath	For type JKS or JCEKS, the absolute path of the keystore file to be imported, including filename. For type OracleWallet, the absolute path of the directory where the Oracle Wallet resides.
permission	Specifies true if keystore is protected by permission only, false if protected by both permission and password.

2.1.3.13.3 Example

This example imports a JKS keystore file to `keystore2`:

```
importKeyStore(appStripe='system', name='keystore2',
password='password',aliases='orakey,seckey', keypasswords='keypassword1,
keypassword2', type='JKS', permission=true, filepath='/tmp/file.jks')
```

This example imports an Oracle Wallet to `keystore2`:

```
importKeyStore(appStripe='system', name='keystore2',
password='mypassword',aliases='orakey,seckey', keypasswords='',
type='OracleWallet', permission=true, filepath='/tmp')
```

2.1.3.14 importKeyStoreCertificate

Imports a certificate or other specified object.

2.1.3.14.1 Description Imports a certificate, trusted certificate or certificate chain.

2.1.3.14.2 Syntax `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.

2.1.3.14.3 Example This example imports a certificate into `keystore2`.

```
importKeyStoreCertificate(appStripe='system', name='keystore2',
password='password', alias='orakey', keypassword='keypassword',
type='Certificate', filepath='/tmp/cert.txt')
```

2.1.3.15 listExpiringCertificates

Lists expiring certificates.

2.1.3.15.1 Description Lists expiring certificates and optionally renews them.

2.1.3.15.2 Syntax `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.

2.1.3.15.3 Example This example lists certificates expiring within one year, and requests that they be renewed:

```
listExpiringCertificates(days='365', autorenew=true)
```

2.1.3.16 listKeyStoreAliases

Lists the aliases in a keystore.

2.1.3.16.1 Description Lists the aliases in a keystore for a given type of entry.

2.1.3.16.2 Syntax The syntax is as follows:

```
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 '*'.

2.1.3.16.3 Example This example lists secret keys in `keystore2`:

```
listKeyStoreAliases(appStripe='system', name='keystore2',
password='password', type='SecretKey')
```

2.1.3.17 listKeyStores

Lists all the keystores in a stripe.

2.1.3.17.1 Description Lists all the keystores in the specified stripe.

2.1.3.17.2 Syntax `listKeyStores(appStripe='stripe')`

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().

Argument	Definition
appStripe	Specifies the name of the stripe whose keystores are listed.

2.1.3.17.3 Example This example lists all keystores on all stripes.

```
listKeyStores(appStripe='*')
```

2.1.3.18 syncKeyStores

Synchronizes Oracle WebLogic Server and system keystores from the central repository to the domain `config` directory on the administration server.

2.1.3.18.1 Description Synchronizes keystores in the central security store with those present in the domain directory.

If the target format is Oracle Wallet, the command synchronizes the contents of all KSS keystores for a given stripe into auto-login wallets on the server.

2.1.3.18.2 Syntax The syntax is as follows:

```
syncKeyStores(stripeName='component-type#component-name',
keystoreFormat='exported_file_format',
rootDirectory='root_dir_absolute_path')
```

Argument	Definition
StripeName	Specifies the name of the stripe corresponding to the component. If `keystoreFormat` is 'OracleWallet', enter the stripe name in the format '*component-type#component-name*'. Keystores in this stripe should be permission-protected only, never password-protected.
keystoreFormat	Specifies the format of the target keystore. Valid formats are 'KSS' and 'OracleWallet'.
rootDirectory	For the Oracle Wallet format, specifies the absolute path of the server directory where the wallet(s) are created. If not specified, defaults to `Admin_Server_Root/config/fmwconfig/`.

Note: The `svc` argument does not apply to this command.

2.1.3.18.3 Example

The following command looks up the central repository for the "system" stripe and downloads its contents into the `keystores.xml` file under the `DOMAIN_HOME/config/fmwconfig` directory. It also downloads the contents of the domain trust store into the same file:

```
syncKeyStores()
```

The following command generates Oracle Wallets corresponding to all keystores in the stripe 'ohs#ohs1':

```
syncKeyStores(stripeName="ohs#ohs1",
keystoreFormat="OracleWallet", rootDirectory="/tmp/bin")
```

2.1.4 Identity Directory Service Commands

Use the WLST commands listed in Table 2–5 to manage Identity Directory Service entity attributes, entity definitions, relationships and default operational configurations.

Table 2–5 WLST Identity Directory Service Commands

Use this command...	To...	Use with WLST...
activateIDSConfigChanges	Reload the Identity Directory Service configuration.	Online
addAttributeInEntityConfig	Add a new attribute to the entity configuration.	Online
addAttributePropsInEntityConfig	Add new properties for an attribute in an entity configuration.	Online
addAttributeRefForEntity	Add a new attribute to the specified entity.	Online
addAttrrefPropsInEntityConfig	Add new properties for an attribute reference in an entity configuration.	Online
addCommonPropertyForOperationConfig	Add a new property for a specified operation configuration.	Online
addEntity	Add a new entity to the entity configuration.	Online
addEntityProps	Add new properties for an entity in an entity configuration.	Online
addEntityRelation	Add a new entity relation to the entity configuration.	Online
addIdentityDirectoryService	Add a new Identity Directory Service to the configuration.	Online
addOperationConfig	Add a new operation configuration to the entity configuration.	Online
addPropertyForOperationConfig	Add a new property to a specified operation configuration.	Online
deleteAttributeInEntityConfig	Delete an attribute from an entity configuration.	Online
deleteAttributePropsInEntityConfig	Delete attribute properties in an entity configuration.	Online
deleteAttrrefPropsInEntityConfig	Delete attribute reference properties in an entity configuration.	Online
deleteEntity	Delete an entity from an entity configuration.	Online
deleteEntityProps	Delete entity properties in an entity configuration.	Online
deleteEntityRelation	Delete the specified entity relation.	Online
deleteIdentityDirectoryService	Delete the specified Identity Directory Service in the configuration.	Online
deleteOperationConfig	Delete operation configuration in an entity configuration.	Online
listAllAttributeInEntityConfig	List all attributes in the entity configuration.	Online
listAllEntityInEntityConfig	List all entities defined in the specified entity configuration.	Online

Table 2–5 (Cont.) WLST Identity Directory Service Commands

Use this command...	To...	Use with WLST...
listAllIdentityDirectoryService	List all Identity Directory Services in the configuration.	Online
removeAttributeRefForEntity	Remove an attribute from the specified entity.	Online
removeCommonPropertyForOperationConfig	Removes a property for the specified operation configuration.	Online
removePropertyForOperationConfig	Remove a property for the specified operation configuration.	Online
updateAttributeInEntityConfig	Update attributes in an entity configuration.	Online
updateAttributePropsInEntityConfig	Update attribute properties in an entity configuration.	Online
updateAttrrefPropsInEntityConfig	Update attribute reference properties in an entity configuration.	Online
updateEntity	Update an entity's properties in an entity configuration.	Online
updateEntityAttrs	Update an entity's properties in an entity configuration.	Online
updateEntityProps	Update the entity properties in an entity configuration.	Online

2.1.4.1 activateIDSConfigChanges

`activateIDSConfigChanges`

2.1.4.1.1 Description Reloads the Identity Directory Service configuration.

2.1.4.1.2 Syntax `activateIDSConfigChanges()`

This command has no arguments.

2.1.4.1.3 Example `activateIDSConfigChanges()`

2.1.4.2 addAttributeInEntityConfig

`addAttributeInEntityConfig`

2.1.4.2.1 Description Adds a new attribute to the entity configuration.

2.1.4.2.2 Syntax `addAttributeInEntityConfig(name, datatype, description, readOnly, pwdAttr, appName)`

Table 2–6 addAttributeInEntityConfig Arguments

Argument	Definition
name	Name of the attribute to be added.

Table 2–6 (Cont.) addAttributeInEntityConfig Arguments

Argument	Definition
datatype	The attribute's type is defined as one of the following: ■ binary ■ boolean ■ datetime ■ double ■ integer ■ rfc822name ■ string ■ x500name
description	Description of the attribute to be added.
readOnly	Flag to specify whether the attribute is read only or can be modified.
pwdAttr	Flag to specify whether the attribute defines a password or not.
appName	Name of the Identity Directory Service.

2.1.4.2.3 Example `addAttributeInEntityConfig('commonname','string','common name',false,false,'userrole')`

2.1.4.3 addAttributePropsInEntityConfig

`addAttributePropsInEntityConfig`

2.1.4.3.1 Description Adds new properties for an attribute in an entity configuration.

2.1.4.3.2 Syntax `addAttributePropsInEntityConfig(name, propNames, propVals, appName)`

Table 2–7 addAttributePropsInEntityConfig Arguments

Argument	Definition
name	Name of the attribute to be added.
propNames	List of property names separated by "|". The properties (*propNames* and *propVals*) are free key/value pairs. Applications can store any required metadata at the attribute level in these properties. The Identity Directory Service does not perform any validation for these property names and does not interpret or use these properties internally. For configuration attributes, however, the Identity Directory Service performs a schema check and interprets the configuration names and their values.
propVals	List of corresponding property values separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.3.3 Example

```
addAttributePropsInEntityConfig('orgunit','labelname|multivalued','common name|true','userrole')
```

2.1.4.4 addAttributeRefForEntity

`addAttributeRefForEntity`

2.1.4.4.1 Description Adds a new attribute to the specified entity.

2.1.4.4.2 Syntax `addAttributeRefForEntity(name, attrRefName, attrRefFilter, attrRefDefaultFetch, appName)`

Table 2–8 addAttributeRefForEntity Arguments

Argument	Definition
name	Name of the entity to which the attribute will be added.
attrRefName	Name of the attribute to be added to the entity.
attrRefFilter	Type of filter to be used with the attribute, defined as one of the following: ■ beginswith ■ contains ■ doesnotcontain ■ dynamic ■ endswith ■ equals ■ greaterequal ■ greaterthan ■ lessequal ■ lessthan ■ none ■ notequals
attrRefDefaultFetch	Flag to specify whether the attribute is fetched by default.
appName	Name of the Identity Directory Service.

2.1.4.4.3 Example

```
addAttributeRefForEntity('User','givenname','none','true','userrole')
```

2.1.4.5 addAttrrefPropsInEntityConfig

`addAttrrefPropsInEntityConfig`

2.1.4.5.1 Description Adds new properties for an attribute reference in an entity configuration.

2.1.4.5.2 Syntax `addAttrrefPropsInEntityConfig(entityName, attrName, propNames, propVals, appName)`

Table 2–9 addAttrrefPropsInEntityConfig Arguments

Argument	Definition
entityName	Name of the entity.
attrName	Name of the attribute reference.

Table 2–9 (Cont.) addAttrrefPropsInEntityConfig Arguments

Argument	Definition
propNames	List of property names separated by "|". The properties (*propNames* and *propVals*) are free key/value pairs. Applications can store any required metadata at the attribute level in these properties. The Identity Directory Service does not perform any validation for these property names and does not interpret or use these properties internally. For configuration attributes, however, the Identity Directory Service performs a schema check and interprets the configuration names and their values.
propVals	List of corresponding property values separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.5.3 Example `addAttrrefPropsInEntityConfig('org', 'orgunit','labelname|multivalued','common name|true','userrole')`

2.1.4.6 addCommonPropertyForOperationConfig

`addCommonPropertyForOperationConfig`

2.1.4.6.1 Description Adds a new property for a specified operation configuration.

2.1.4.6.2 Syntax `addCommonPropertyForOperationConfig(entityName, propName, propValue, appName)`

Table 2–10 addCommonPropertyForOperationConfig Arguments

Argument	Definition
entityName	Name of the entity.
propName	Name of the property to be added for this operation configuration.
propValue	Value of the property to be added for this operation configuration.
appName	Name of the Identity Directory Service.

2.1.4.6.3 Example `addCommonPropertyForOperationConfig('groupmember.attr', 'member', 'userrole')`

2.1.4.7 addEntity

`addEntity`

2.1.4.7.1 Description Adds a new entity to the entity configuration.

2.1.4.7.2 Syntax `addEntity(name, type, idAttr, create, modify, delete, search, attrRefNames, attrRefFilters, attrRefDefaultFetches, appName)`

Table 2–11 addEntity Arguments

Argument	Definition
name	Name of the entity to which the attribute will be added.
type	Name of the attribute to be added to the entity.

Table 2–11 (Cont.) addEntity Arguments

Argument	Definition
idAttr	Identity attribute of the entity to be added.
create	Flag to specify the create is allowed.
modify	Flag to specify the modify is allowed.
delete	Flag to specify the delete is allowed.
search	Flag to specify the search is allowed.
attrRefNames	Array of attribute names.
attrRefFilters	An array of filter type values, defined as one of the following: ■ beginswith ■ contains ■ doesnotcontain ■ dynamic ■ endswith ■ equals ■ greaterequal ■ greaterthan ■ lessequal ■ lessthan ■ none ■ notequals
attrRefDefaultFetches	Array of boolean strings (true, false).
appName	Name of the Identity Directory Service.

2.1.4.7.3 Example

```
addEntity('Group','group','commonname',true,true,true,true,'name|commonnam
e','none|none','true|false','userrole')
```

2.1.4.8 addEntityProps

```
addEntityProps
```

2.1.4.8.1 Description Adds new properties for an entity in an entity configuration.

2.1.4.8.2 Syntax `addEntityProps(name, propNames, propVals, appName)`

Table 2–12 addEntityProps Arguments

Argument	Definition
name	Name of the entity.
propNames	List of property names separated by "|".
propValues	List of corresponding property values separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.8.3 Example

```
addEntityProps('User','inclobjclasses|exclobjclasses','inetorgperson|orcli
dxperson','userrole')
```

2.1.4.9 addEntityRelation

`addEntityRelation`

2.1.4.9.1 Description Add a new entity relation to the entity configuration.

2.1.4.9.2 Syntax `addEntityRelation(name, type, fromEntity, fromAttr, toEntity, toAttr, recursive, appName)`

Table 2–13 addEntityRelation Arguments

Argument	Definition
name	Name of the relation between the entities for the given attributes.
type	Type of the entity relation ("ManyToMany", "ManyToOne", "OneToMany", "OneToOne").
fromEntity	Name of the from entity.
fromAttr	Name of the from attribute.
toEntity	Name of the to entity.
toAttr	Name of the to attribute.
recursive	Flag to set the entity relationship as recursive.
appName	Name of the Identity Directory Service.

2.1.4.9.3 Example `addEntityRelation('manager', 'ManyToOne', 'User', 'manager', 'User', 'principal', false, 'userrole')`

2.1.4.10 addIdentityDirectoryService

`addIdentityDirectoryService`

2.1.4.10.1 Description Adds a new IdentityStoreService to the Identity Directory Service configuration.

2.1.4.10.2 Syntax `addIdentityDirectoryService(name, description, propNames, propValues)`

Table 2–14 addIdentityDirectoryService Arguments

Argument	Definition
name	Name of the IdentityStoreService to be added.
description	Description of the IdentityStoreService.
propNames	An array of property names to be added to the IdentityStoreService configuration.
propValues	An array of values to be defined for the property names added to the IdentityStoreService configuration.

2.1.4.10.3 Example `addIdentityDirectoryService('userrole', 'user role', 'ovd.context|entity.config', 'default|userrole')`

2.1.4.11 addOperationConfig

`addOperationConfig`

2.1.4.11.1 Description Adds a new operation configuration to the entity configuration.

2.1.4.11.2 Syntax `addOperationConfig(entityName, propNames, propValues, appName)`

Table 2–15 addOperationConfig Arguments

Argument	Definition
entityName	Name of the entity to which the operation configuration will be added.
propNames	An array of property names to be added to the operation configuration.
propValues	An array of property values for the properties added to the operation configuration.
appName	Name of the Identity Directory Service.

2.1.4.11.3 Example `addOperationConfig('User', 'entity.searchbase', 'cn=users,dc=oracle,dc=com', 'userrole')`

2.1.4.12 addPropertyForOperationConfig

`addPropertyForOperationConfig`

2.1.4.12.1 Description Adds a new property to a specified operation configuration.

2.1.4.12.2 Syntax `addPropertyForOperationConfig(entityName, propName, propValue, appName)`

Table 2–16 addPropertyForOperationConfig Arguments

Argument	Definition
entityName	Name of the entity to which the operation configuration will be added.
propName	A property name to be added to the operation configuration.
propValue	A value for the property added to the operation configuration.
appName	Name of the Identity Directory Service.

2.1.4.12.3 Example `addPropertyForOperationConfig('User','entity.searchbase', 'cn=users,dc=oracle,dc=com', 'userrole')`

2.1.4.13 deleteAttributeInEntityConfig

`deleteAttributeInEntityConfig`

2.1.4.13.1 Description Deletes an attribute from an entity configuration.

2.1.4.13.2 Syntax `deleteAttributeInEntityConfig(name, appName)`

Table 2–17 deleteAttributeInEntityConfig Arguments

Argument	Definition
name	Name of the attribute to be deleted.
appName	Name of the Identity Directory Service.

2.1.4.13.3 Example `deleteAttributeInEntityConfig('commonname', 'userrole')`

2.1.4.14 deleteAttributePropsInEntityConfig

`deleteAttributePropsInEntityConfig`

2.1.4.14.1 Description Deletes attribute properties in an entity configuration.

2.1.4.14.2 Syntax `deleteAttributePropsInEntityConfig(name, propNames, appName)`

Table 2–18 deleteAttributePropsInEntityConfig Arguments

Argument	Definition
name	Name of the attribute.
propNames	List of property names separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.14.3 Example

```
deleteAttributePropsInEntityConfig('orgunit','labelname|multivalued','user
role')
```

2.1.4.15 deleteAttrrefPropsInEntityConfig

`deleteAttrrefPropsInEntityConfig`

2.1.4.15.1 Description Deletes attribute reference properties in an entity configuration.

2.1.4.15.2 Syntax `deleteAttrrefPropsInEntityConfig(entityName, attrName, propNames, appName)`

Table 2–19 deleteAttrrefPropsInEntityConfig Arguments

Argument	Definition
entityName	Name of the entity.
attrName	Name of the attribute reference.
propNames	List of property names to be deleted. If multiple properties are to be deleted, they should be separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.15.3 Example `deleteAttrrefPropsInEntityConfig('org', 'orgunit','labelname|multivalued','userrole')`

2.1.4.16 deleteEntity

`deleteEntity`

2.1.4.16.1 Description Deletes an entity from an entity configuration.

2.1.4.16.2 Syntax `deleteEntity(name, appName)`

Table 2–20 deleteEntity Arguments

Argument	Definition
name	Name of the entity to be deleted.
appName	Name of the Identity Directory Service.

2.1.4.16.3 Example `deleteEntity('User', 'userrole')`

2.1.4.17 deleteEntityProps

`deleteEntityProps`

2.1.4.17.1 Description Deletes entity properties in an entity configuration.

2.1.4.17.2 Syntax `deleteEntityProps(name, propNames, appName)`

Table 2–21 deleteEntityProps Arguments

Argument	Definition
name	Name of the entity.
propNames	List of property names separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.17.3 Example

```
deleteEntityProps('User','inclobjclasses|exclobjclasses','userrole')
```

2.1.4.18 deleteEntityRelation

`deleteEntityRelation`

2.1.4.18.1 Description Deletes the specified entity relation.

2.1.4.18.2 Syntax `deleteEntityRelation(name, appName)`

Table 2–22 deleteEntityRelation Arguments

Argument	Definition
name	Name of the relation between the entities for the given attributes.
appName	Name of the Identity Directory Service.

2.1.4.18.3 Example `deleteEntityRelation('manager', 'userrole')`

2.1.4.19 deleteIdentityDirectoryService

`deleteIdentityDirectoryService'`

2.1.4.19.1 Description Deletes the specified IdentityStoreService in the Identity Directory Service configuration.

2.1.4.19.2 Syntax `deleteIdentityDirectoryService(name)`

where *name* is the name of the IdentityStoreService configuration to be deleted.

2.1.4.19.3 Example `deleteIdentityDirectoryService('ids1')`

2.1.4.20 deleteOperationConfig

`deleteOperationConfig`

2.1.4.20.1 Description Deletes an operation configuration in an entity configuration.

2.1.4.20.2 Syntax `deleteOperationConfig(entityName, appName)`

Table 2–23 deleteOperationConfig Arguments

Argument	Definition
entityName	Name of the entity from which the operation configuration will be removed.
appName	Name of the Identity Directory Service.

2.1.4.20.3 Example `deleteOperationConfig('User','userrole')`

2.1.4.21 listAllAttributeInEntityConfig

`listAllAttributeInEntityConfig`

2.1.4.21.1 Description Lists all attributes in the entity configuration.

2.1.4.21.2 Syntax `listAllAttributeInEntityConfig(appName)`

where *appName* is the name of the Identity Directory Service that contains the entity configuration from which the list of attributes is retrieved.

2.1.4.21.3 Example `listAllAttributeInEntityConfig('userrole')`

2.1.4.22 listAllEntityInEntityConfig

`listAllEntityInEntityConfig`

2.1.4.22.1 Description Lists all entities defined in the specified entity configuration.

2.1.4.22.2 Syntax `listAllEntityInEntityConfig(appName)`

where *appName* is the name of the Identity Directory Service that contains the entity configuration from which the list of entities is retrieved.

2.1.4.22.3 Example `listAllEntityInEntityConfig('userrole')`

2.1.4.23 listAllIdentityDirectoryService

`listAllIdentityDirectoryService`

2.1.4.23.1 Description Lists all IdentityStoreService in Identity Directory Service configuration.

2.1.4.23.2 Syntax `listAllIdentityDirectoryService()`

This command has no arguments.

2.1.4.23.3 Example `listAllIdentityDirectoryService()`

2.1.4.24 removeAttributeRefForEntity

`removeAttributeRefForEntity`

2.1.4.24.1 Description Removes an attribute from the specified entity.

2.1.4.24.2 Syntax `removeAttributeRefForEntity(name, attrRefName, appName)`

Table 2–24 removeAttributeRefForEntity Arguments

Argument	Definition
name	Name of the entity from which the attribute will be removed.
attrRefName	The name of the attribute to be removed.
appName	Name of the Identity Directory Service.

2.1.4.24.3 Example `removeAttributeRefForEntity('User','givenname','userrole')`

2.1.4.25 removeCommonPropertyForOperationConfig

`removeCommonPropertyForOperationConfig`

2.1.4.25.1 Description Removes a property for the specified operation configuration.

2.1.4.25.2 Syntax `removeCommonPropertyForOperationConfig(entityName, propName, appName)`

Table 2–25 removeCommonPropertyForOperationConfig Arguments

Argument	Definition
entityName	Name of the entity.
propName	Name of property to be removed for this operation configuration.
appName	Name of the Identity Directory Service.

2.1.4.25.3 Example

```
removeCommonPropertyForOperationConfig('groupmember.attr','userrole')
```

2.1.4.26 removePropertyForOperationConfig

`removePropertyForOperationConfig`

2.1.4.26.1 Description Removes a property for the specified operation configuration.

2.1.4.26.2 Syntax `removePropertyForOperationConfig(entityName, propName, appName)`

Table 2–26 removePropertyForOperationConfig Arguments

Argument	Definition
entityName	Name of the entity to which the operation configuration will be added.
propName	A property name to be added to the operation configuration.
appName	Name of the Identity Directory Service.

2.1.4.26.3 Example

```
removePropertyForOperationConfig('User','entity.searchbase','userrole')
```

2.1.4.27 updateAttributeInEntityConfig

`updateAttributeInEntityConfig`

2.1.4.27.1 Description Updates attributes in an entity configuration.

2.1.4.27.2 Syntax `updateAttributeInEntityConfig(name, attrNames, attrVals, appName)`

Table 2–27 updateAttributeInEntityConfig Arguments

Argument	Definition
name	Name of the entity attribute to be updated.
attrNames	List of configuration attribute names separated by "|". Valid configuration attribute names are: ■ `dataType` ■ `description` ■ `readOnly` ■ `pwdAttr` ■ `attrInUse`
attrVals	List of corresponding attribute values separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.27.3 Example

```
updateAttributeInEntityConfig('commonname','readOnly|pwdAttr|attrInUse','true|false|false','userrole')
```

2.1.4.28 updateAttributePropsInEntityConfig

`updateAttributePropsInEntityConfig`

2.1.4.28.1 Description Updates attribute properties in an entity configuration.

2.1.4.28.2 Syntax `updateAttributePropsInEntityConfig(name, propNames, propVals, appName)`

Table 2–28 updateAttributePropsInEntityConfig Arguments

Argument	Definition
name	Name of the attribute to be updated.
propNames	List of property names separated by "|".
propVals	List of corresponding property values separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.28.3 Example

```
updateAttributePropsInEntityConfig('orgunit','multivalued','multivalued','userrole')
```

2.1.4.29 updateAttrrefPropsInEntityConfig

`updateAttrrefPropsInEntityConfig`

2.1.4.29.1 Description Updates attribute reference properties in an entity configuration.

2.1.4.29.2 Syntax `updateAttrrefPropsInEntityConfig(entityName, attrName, propNames, propVals, appName)`

Table 2–29 updateAttrrefPropsInEntityConfig Arguments

Argument	Definition
entityName	Name of the entity.
attrName	Name of the attribute reference.
propNames	List of property names separated by "|".
propVals	List of corresponding property values separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.29.3 Example `updateAttrrefPropsInEntityConfig('org', 'orgunit','multivalued','multivalued','userrole')`

2.1.4.30 updateEntity

`updateEntity`

2.1.4.30.1 Description Updates an entity's properties in an entity configuration.

2.1.4.30.2 Syntax `updateEntity(name, type, idAttr, create, modify, delete, search, appName)`

Table 2–30 updateEntity Arguments

Argument	Definition
name	Name of the entity to be updated.
type	Type of the entity.
idAttr	Identity attribute of the entity.
create	Flag to specify the create is allowed.
modify	Flag to specify the modify is allowed.
delete	Flag to specify the delete is allowed.
search	Flag to specify the search is allowed.
appName	Name of the Identity Directory Service.

2.1.4.30.3 Example

```
updateEntity('Group','group','commonname',true,true,true,true,'userrole')
```

2.1.4.31 updateEntityAttrs

`updateEntityAttrs`

2.1.4.31.1 Description Updates the configuration attributes for an entity attribute.

2.1.4.31.2 Syntax `updateEntityAttrs(name, attrNames, attrVals, appName)`

Table 2–31 updateEntityAttrs Arguments

Argument	Definition
name	Name of the entity attribute. To update the properties of an entity attribute, see updateAttributePropsInEntityConfig.

Table 2–31 (Cont.) updateEntityAttrs Arguments

Argument	Definition
attrNames	List of configuration attribute names. If multiple configuration attributes are to be updated, they should be separated by "|". Valid configuration attribute names are: ■ idAttr ■ pwdAttr ■ firstnameAttr ■ lastnameAttr ■ mailAttr ■ displaynameAttr ■ descriptionAttr ■ challengeQnAttr ■ challengeAnsAttr ■ commonIdAttr.
attrVals	List of corresponding configuration attribute values separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.31.3 Example

```
updateEntityAttrs('User','idAttr|firstnameAttr','uid|givenname','userrole'
)
```

2.1.4.32 updateEntityProps

```
updateEntityProps
```

2.1.4.32.1 Description Updates the entity properties in an entity configuration.

2.1.4.32.2 Syntax `updateEntityProps(name, propNames, propVals, appName)`

Table 2–32 updateEntityProps Arguments

Argument	Definition
name	Name of the attribute to be added.
propNames	List of property names separated by "|".
propVals	List of corresponding property values separated by "|".
appName	Name of the Identity Directory Service.

2.1.4.32.3 Example

```
updateEntityProps('User','inclobjclasses|exclobjclasses','inetorgperson|or
clidxperson','userrole')
```

2.1.5 Library Oracle Virtual Directory (libOVD) Commands

Use the WLST commands listed in Table 2–33 to manage a libOVD configuration associated with a specific Oracle Platform Security Services (OPSS) context.

Table 2–33 WLST libOVD Commands

Use this command...	To...	Use with WLST...
addDNAttribute	Add an attribute to the DN attributes list for an existing adapter.	Online
activateLibOVDConfigChanges	Reload the libOVD configuration.	Online
addAttributeExclusionRule	Add a attribute exclusion rule.	Online
addAttributeRule	Add a new attribute mapping rule.	Online
addDomainExclusionRule	Add a domain exclusion rule.	Online
addDomainRule	Add a new domain mapping rule.	Online
addJoinRule	Add a join rule to an existing Join Adapter for a libOVD configuration.	Online
addLDAPHost	Add a new remote host to an existing LDAP adapter.	Online
addMappingContext	Create a new mapping context.	Online
addPlugin	Add a plug-in to an existing adapter or at the global level.	Online
addPluginParam	Add new parameter values to the existing adapter level plug-in or global plug-in.	Online
addToRequestControlExcludeList	Add a control to the Request Control Exclude List for an existing LDAP adapter configuration.	Online
addToRequestControlIncludeList	Add a control to the Request Control Include List for an existing LDAP adapter configuration.	Online
assignViewToAdapter	Assign the given view to an adapter.	Online
createJoinAdapter	Create a new Join Adapter for a libOVD configuration.	Online
createLDAPAdapter	Create a new LDAP adapter for a libOVD configuration.	Online
createLDAPAdapterWithDefaultPlugins	Create a new LDAP adapter with default plug-ins based on the specified directory type.	Online
createView	Create a new view.	Online
deleteAdapter	Delete an existing adapter for a libOVD configuration.	Online
deleteAttributeExclusionRule	Delete a attribute exclusion rule.	Online
deleteAttributeRule	Delete a attribute mapping rule.	Online
deleteDomainExclusionRule	Delete a domain exclusion rule.	Online
deleteDomainRule	Delete a domain mapping rule.	Online
deleteMappingContext	Delete the specified mapping context.	Online
deleteView	Delete the specified view.	Online
getAdapterDetails	Display the details of an existing adapter for a libOVD configuration.	Online
listAdapters	List the name and type of all adapters that are configured for a libOVD configuration.	Online
listAllMappingContextIds	List all the mapping contexts.	Online
listAttributeRules	List all the attribute rules.	Online
listDomainRules	List all the domain rules.	Online
listViews	List all views	Online
modifyLDAPAdapter	Modify the existing LDAP adapter configuration.	Online
modifySocketOptions	Modify the socket options for an existing LDAP adapter configuration.	Online
removeAllRequestControlExcludeList	Remove all controls from the Request Control Exclude List for an existing LDAP adapter configuration.	Online

Table 2–33 (Cont.) WLST libOVD Commands

Use this command...	To...	Use with WLST...
removeAllRequestControlIncludeList	Remove all controls from a Request Control Include List for an existing LDAP adapter configuration.	Online
removeDNAttribute	Remove an attribute from the DN attributes list for an existing LDAP adapter configuration.	Online
removeFromRequestControlExcludeList	Remove a control from the Request Control Exclude List for an existing LDAP adapter configuration.	Online
removeFromRequestControlIncludeList	Removes a control from the Request Control Include List for an existing LDAP adapter configuration.	Online
removeJoinRule	Remove a join rule from a Join Adapter configured for a libOVD configuration.	Online
removeLDAPHost	Remove a remote host from an existing LDAP adapter configuration.	Online
removePlugin	Remove a plug-in from an existing adapter or at the global level.	Online
removePluginParam	Remove an existing parameter from a configured adapter level plug-in or global plug-in.	Online
replacePluginParam	Replace existing parameter values for an adapter level plug-in or global plug-in.	Online
unassignViewFromAdapter	Unassign a view from an adapter.	Online

2.1.5.1 addDNAttribute

Adds an attribute to the DN Attributes List.

2.1.5.1.1 Description Adds an attribute to the DN Attributes List for an existing adapter configured for the libOVD configuration associated with an OPSS context.

2.1.5.1.2 Syntax `addDNAttribute(adapterName, attributeName, [contextName])`

Table 2–34 addDNAttribute Arguments

Argument	Definition
adapterName	Name of the adapter to be updated.
attributeName	Name of the new DN attribute to be added.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.1.3 Example

```
addDNAttribute(adapterName='ldap1', attributeName='memberof',
contextName='default')
```

2.1.5.2 activateLibOVDConfigChanges

Reloads the libOVD configuration.

2.1.5.2.1 Description Reloads the libOVD configuration associated with a specific OPSS context.

2.1.5.2.2 Syntax `activateLibOVDConfigChanges([contextName])`

Table 2–35 activateLibOVDConfigChanges Arguments

Argument	Definition
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.2.3 Example

```
activateLibOVDConfigChanges('default')
```

2.1.5.3 addAttributeExclusionRule

Adds an attribute exclusion rule.

2.1.5.3.1 Description Adds an attribute exclusion rule to the exclusion list.

2.1.5.3.2 Syntax `addAttributeExclusionRule(attribute, mappingContextId, [contextName])`

Table 2–36 addAttributeExclusionRule Arguments

Argument	Definition
attribute	Name of the attribute to be added to the exclusion list.
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.3.3 Example

```
addAttributeExclusionRule('objectsid')
```

2.1.5.4 addAttributeRule

Adds a new attribute mapping rule.

2.1.5.4.1 Description Adds a new attribute mapping rule to the libOVD configuration associated with a specific OPSS context..

2.1.5.4.2 Syntax `addAttributeRule(srcAttrs, srcObjectClass, srcAttrType, dstAttr, dstObjectClass, dstAttrType, mappingExpression, direction, mappingContextId, [contextName])`

Table 2–37 addAttributeRule Arguments

Argument	Definition
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.4.3 Example

```
addAttributeRule('lastname','','','sn','','','','Inbound')
```

2.1.5.5 addDomainExclusionRule

Adds a domain exclusion rule.

2.1.5.5.1 Description Adds a domain exclusion rule to the exclusion list.

2.1.5.5.2 Syntax `addDomainExclusionRule(domain, mappingContextId, [contextName])`

Table 2–38 addDomainExclusionRule Arguments

Argument	Definition
domain	Distinguished name (DN) of the attribute to be added to the exclusion list.
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.5.3 Example

```
addDomainExclusionRule('cn=group,dc=oracle,dc=com')
```

2.1.5.6 addDomainRule

Adds a new domain mapping rule.

2.1.5.6.1 Description Adds a new domain mapping rule.

2.1.5.6.2 Syntax `addDoma]inRule(srcDomain, destDomain, domainConstructRule, mappingContextId, [contextName])`

Table 2–39 addDomainRule Arguments

Argument	Definition
srcDomain	Source domain.
destDomain	Destination domain
domainConstructRule	Name of the attribute to be added to the exclusion list.
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.6.3 Example

```
addDomainRule('dc=oracle,dc=com', 'dc=oracle,dc=com', '', 'defaultContext',
'default')
```

2.1.5.7 addJoinRule

Adds a join rule to a Join Adapter.

2.1.5.7.1 Description Adds a join rule to an existing Join Adapter for the libOVD configuration associated with the specified OPSS context.

2.1.5.7.2 Syntax `addJoinRule(adapterName, secondary, condition, [joinerType], [contextName])`

Table 2–40 addJoinRule Arguments

Argument	Definition
adapterName	Name of the Join Adapter to be modified.
secondary	Name of the adapter to join to.
condition	The attribute(s) to join on.
joinerType	Optional. Defines the type of Join. Values can be Simple (default), Conditional, OneToMany, or Shadow.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.7.3 Examples

```
addJoinRule('join1','secondaryldap','cn=cn', 'Simple', 'default')

addJoinRule('join1','secondaryldap','cn=cn', 'Conditional', 'default')

addJoinRule(adapterName='join1', secondary='LDAP3', condition='uid=cn',
JoinerType='OneToMany')

addJoinRule(adapterName='join1', secondary='LDAP2',condition='uid=cn',
contextName='myContext')
```

2.1.5.8 addLDAPHost

Adds a new remote host.

2.1.5.8.1 Description Adds a new remote host (host and port) to an existing LDAP adapter. By default, the new host is configured in Read-Write mode with percentage set to 100.

2.1.5.8.2 Syntax `addLDAPHost(adapterName, host, port, [contextName])`

Table 2–41 addLDAPHost Arguments

Argument	Definition
adapterName	Name of the Join Adapter to be modified.
host	Remote LDAP host to which the LDAP adapter will communicate.
port	Remote LDAP host port.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.8.3 Examples

```
addLDAPHost(adapterName='ldap1', host='myhost.example.com', port=389)

addLDAPHost('ldap1', 'myhost.example.com','389', 'myContext')
```

2.1.5.9 addMappingContext

Creates a new mapping context.

2.1.5.9.1 Description Creates a new mapping context for the libOVD configuration associated with the specified OPSS context.

2.1.5.9.2 Syntax `addMappingContext(mappingContextId, [contextName])`

Table 2–42 addMappingContext Arguments

Argument	Definition
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.9.3 Example

```
addMappingContext('defaultContext','context')
```

2.1.5.10 addPlugin

Adds a plug-in to an existing adapter or at the global level.

2.1.5.10.1 Description Adds a plug-in to an existing adapter or at the global level. The "i"th key corresponds to "i"th value. The plug-in is added to default chain.

2.1.5.10.2 Syntax `addPlugin(pluginName, pluginClass, paramKeys, paramValues, [adapterName], [contextName])`

Table 2–43 addPlugin Arguments

Argument	Definition
pluginName	Name of the plug-in to be created.
pluginClass	Class of the plug-in.
paramKeys	Init Param Keys separated by "|".
paramValues	Init Param Values separated by "|".
adapterName	Optional. Name of the adapter to be modified. If not specified, the plug-in is added at the global level.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.10.3 Examples

```
addPlugin(adapterName='ldap1',
pluginName='VirtualAttr',pluginClass='oracle.ods.virtualization.engine.chain.plugi
ns.virtualattr.VirtualAttributePlugin', paramKeys='AddAttribute | MatchFilter |
ContainerDN', paramValues='cn=%uid% | objectclass=person | dc=oracle,dc=com')

addPlugin(pluginName='VirtualAttr',pluginClass='oracle.ods.virtualization.engine.c
hain.plugins.virtualattr.VirtualAttributePlugin', paramKeys='AddAttribute |
MatchFilter | ContainerDN', paramValues='cn=%uid% | objectclass=person |
dc=oracle,dc=com')
```

```
addPlugin(pluginName='DMSMetrics',pluginClass='oracle.ods.virtualization.engine.ch
ain.plugins.DMSMetrics.MonitorPerformance',
paramKeys='None',paramValues='None',adapterName='ldap1',contextName='default')
```

2.1.5.11 addPluginParam

Adds new parameter values to the existing adapter level plug-in or global plug-in.

2.1.5.11.1 Description Adds new parameter values to the existing adapter level plug-in or the global plug-in. If the parameter already exists, the new value is added to the existing set of values. The "i"th key corresponds to "i"th value.

2.1.5.11.2 Syntax `addPluginParam(pluginName, paramKeys, paramValues, [adapterName], [contextName])`

Table 2–44 addPluginParam Arguments

Argument	Definition
pluginName	Name of the plug-in to be modified.
paramKeys	Init Param Keys separated by "|".
paramValues	Init Param Values separated by "|".
adapterName	Optional Name of the adapter to be modified. If not specified, the global plug-in is modified.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.11.3 Examples

```
addPluginParam(adapterName='ldap1', pluginName='VirtualAttr',
paramKeys='ReplaceAttribute | MatchFilter', paramValues='cn=%uid% |
objectclass=person')

addPluginParam(pluginName='VirtualAttr', paramKeys='ReplaceAttribute |
MatchFilter', par)
```

2.1.5.12 addToRequestControlExcludeList

Adds a control to the Request Control Exclude List.

2.1.5.12.1 Description Adds a control to the Request Control Exclude List for an existing LDAP adapter configuration.

2.1.5.12.2 Syntax `addToRequestControlExcludeList(adapterName, control, [contextName])`

Table 2–45 addToRequestControlExcludeList Arguments

Argument	Definition
adapterName	Name of the LDAP adapter to be modified.
control	LDAP control object identifier (OID).

Table 2–45 (Cont.) addToRequestControlExcludeList Arguments

Argument	Definition
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.12.3 Example

```
addToRequestControlExcludeList(adapterName='ldap1',
control='2.16.840.1.113894.1.8.31', contextName='default')
```

2.1.5.13 addToRequestControlIncludeList

Adds a control to the Request Control Include List.

2.1.5.13.1 Description Adds a control to the Request Control Include List for an existing LDAP adapter configuration.

2.1.5.13.2 Syntax `addToRequestControlIncludeList(adapterName, control, [contextName])`

Table 2–46 addToRequestControlIncludeList Arguments

Argument	Definition
adapterName	Name of the LDAP adapter to be modified.
control	LDAP control object identifier (OID).
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.13.3 Example

```
addToRequestControlIncludeList(adapterName='ldap1',
control='2.16.840.1.113894.1.8.31', contextName='default')
```

2.1.5.14 assignViewToAdapter

Assigns a view to an LDAP adapter.

2.1.5.14.1 Description Assigns a view to an LDAP adapter in the libOVD configuration associated with an OPSS context.

2.1.5.14.2 Syntax `assignViewToAdapter(viewName, adapterName, [contextName])`

Table 2–47 assignViewToAdapter Arguments

Argument	Definition
viewName	Name of the view.
adapterName	Name of the LDAP adapter.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.14.3 Example

```
assignViewToAdapter('userView','ldap1', 'default')
```

2.1.5.15 createJoinAdapter

Creates a new Join Adapter.

2.1.5.15.1 Description Creates a new Join Adapter for the libOVD configuration associated with an OPSS context.

2.1.5.15.2 Syntax `createJoinAdapter([contextName], adapterName, root, primaryAdapter, bindAdapter)`

Table 2–48 createJoinAdapter Arguments

Argument	Definition
adapterName	Name of the Join Adapter to be created.
mappingContextId	Virtual Namespace of the Join Adapter.
primaryAdapter	Specifies the identifier of the primary adapter, which is the adapter searched first in the join operation.
root	root
bindAdapter	Specifies identifier of the bind adapter(s), which are the adapter(s) whose proxy account is used to bind in the LDAP operation. By default, `primaryAdapter` is set as `bindAdapter`.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.15.3 Examples

```
createJoinAdapter('join1','dc=join','primaryldap','myldap', 'myContext')

createJoinAdapter(adapterName='join1', root='dc=join', primaryAdapter='myldap')
```

2.1.5.16 createLDAPAdapter

Creates a new LDAP adapter.

2.1.5.16.1 Description Creates a new LDAP adapter for the libOVD configuration associated with an OPSS context.

2.1.5.16.2 Syntax `createLDAPAdapter(adapterName, root, host, port, remoteBase, [isSecure], [bindDN], [bindPasswd], [passCred], [contextName])`

Table 2–49 createLDAPAdapter Arguments

Argument	Definition
adapterName	Name of the LDAP adapter to be created.
root	Virtual Namespace of the LDAP adapter.
host	Remote LDAP host with which the LDAP adapter will communicate.
port	Remote LDAP host port number.

Table 2–49 (Cont.) createLDAPAdapter Arguments

Argument	Definition
remoteBase	Location in the remote DIT to which root corresponds.
isSecure	Optional. Boolean value that enables secure SSL/TLS connections to the remote hosts when set to `true`. The default value is `false`.
bindDN	Optional. Proxy `BindDN` used to communicate with remote host. Default value is "".
bindPasswd	Optional. Proxy `BindPasswd` used to communicate with the remote host. Default value is "".
passCred	Optional. Controls the credentials, if any, the libOVD configuration will pass to the back-end (remote host) LDAP server. Values can be Always (default), None, or BindOnly.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.16.3 Examples

```
createLDAPAdapter("testLDAP", "dc=us,dc=oracle,dc=com", "myhost.example.com",
3060, "dc=uk,dc=oid", false, "cn=testuser", "welcome1", "Always", "myContext"

createLDAPAdapter(adapterName='ldap1', root='dc=com', host='myhost.example.com',
port=5566, remoteBase='dc=oid')
```

2.1.5.17 createLDAPAdapterWithDefaultPlugins

Creates a new LDAP adapter.

2.1.5.17.1 Description Creates a new LDAP adapter with default plug-ins based on the directory type for the libOVD configuration associated with an OPSS context.

2.1.5.17.2 Syntax `createLDAPAdapterWithDefaultPlugins(adapterName, directoryType, root, host, port, remoteBase, [isSecure], [bindDN], [bindPasswd], [contextName])`

Table 2–50 createLDAPAdapterWithDefaultPlugins Arguments

Argument	Definition
adapterName	Name of the LDAP adapter to be created.
directoryType	Directory type. The value can be one of the following directories: ■ OID - Oracle Internet Directory ■ OUD - Oracle Unified Directory ■ SUNONE- Sun Java System Directory Server ■ OVD - Oracle Virtual Directory ■ ACTIVE_DIRECTORY - Microsoft Active Directory ■ EDIRECTORY - Novell eDirectory ■ OPEN_LDAP - Open LDAP ■ WLS_OVD - Oracle WebLogic Server OVD ■ TIVOLI - IBM Tivoli Directory Server
root	Virtual Namespace of the LDAP adapter.
host	Remote LDAP host to which LDAP adapter should communicate.

Table 2–50 (Cont.) createLDAPAdapterWithDefaultPlugins Arguments

Argument	Definition
port	Remote host port.
remoteBase	Location in the remote DIT to which the root corresponds.
isSecure	Optional. Boolean value that enables secure SSL/TLS connections to the remote hosts when set to `true`. The default value is `false`.
bindDN	Optional. Proxy `BindDN` used to communicate with remote host. Default value is "".
bindPasswd	Optional. Proxy `BindPasswd` used to communicate with the remote host. Default value is "".
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.17.3 Examples

```
createLDAPAdapterWithDefaultPlugins("testLDAP", "OID", "dc=us,dc=oracle,dc=com",
"myhost.example.domain.com", 3060, "dc=uk,dc=oid", false, "cn=testuser",
"welcome1", "myContext")

createLDAPAdapterWithDefaultPlugins(adapterName='ldap1', directoryType="OID",
root='dc=com', host='myhost.example.domain.com', port=5566,
remoteBase='dc=oid',bindDN="cn=testuser",bindPasswd="welcome1",contextName='defaul
t')
```

2.1.5.18 createView

Creates a new view.

2.1.5.18.1 Description Creates a new view for the libOVD configuration associated with an OPSS context.

2.1.5.18.2 Syntax `createView(viewName, [contextName])`

Table 2–51 createView Arguments

Argument	Definition
viewName	Name of the new view.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.18.3 Example

```
createView('userView','default')
```

2.1.5.19 deleteAdapter

Deletes an existing adapter.

2.1.5.19.1 Description Deletes an existing adapter for the libOVD configuration associated with an OPSS context.

2.1.5.19.2 Syntax `deleteAdapter(adapterName, [contextName])`

Table 2–52 deleteAdapter Arguments

Argument	Definition
adapterName	Name of the Join Adapter to be deleted.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.19.3 Examples

```
deleteAdapter(adapterName='join1') deleteAdapter('join1', 'default'))
```

2.1.5.20 deleteAttributeExclusionRule

Deletes an attribute exclusion rule.

2.1.5.20.1 Description Deletes an attribute exclusion rule for the libOVD configuration associated with an OPSS context.

2.1.5.20.2 Syntax `deleteAttributeExclusionRule(attribute, mappingContextId, [contextName])`

Table 2–53 deleteAttributeExclusionRule Arguments

Argument	Definition
attribute	Name of the attribute to be removed from the exclusion list.
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.20.3 Example

```
deleteAttributeExclusionRule('objectsid')
```

2.1.5.21 deleteAttributeRule

Delete an attribute mapping rule.

2.1.5.21.1 Description Delete an attribute mapping rule for the libOVD configuration associated with an OPSS context.

2.1.5.21.2 Syntax `deleteAttributeRule(srcAttrs, dstAttr, mappingContextId, [contextName])`

Table 2–54 deleteEntityRelation Arguments

Argument	Definition
srcAttrs	Source attributes.
dstAttr	Destination attribute.
mappingContextId	Name of the mapping context.

Table 2–54 (Cont.) deleteEntityRelation Arguments

Argument	Definition
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.21.3 Example

```
deleteAttributeRule('lastname','sn')
```

2.1.5.22 deleteDomainExclusionRule

Deletes a domain exclusion rule.

2.1.5.22.1 Description Deletes a domain exclusion rule for the libOVD configuration associated with an OPSS context.

2.1.5.22.2 Syntax `deleteDomainExclusionRule(domain, mappingContextId, [contextName])`

Table 2–55 deleteEntityRelation Arguments

Argument	Definition
domain	Distinguished Name of the container to be removed from the exclusion list.
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.22.3 Example

```
deleteDomainExclusionRule('cn=group,dc=oracle,dc=com')
```

2.1.5.23 deleteDomainRule

Deletes a domain mapping rule.

2.1.5.23.1 Description Deletes a domain mapping rule for the libOVD configuration associated with an OPSS context.

2.1.5.23.2 Syntax `deleteDomainRule(srcDomain, destDomain, mappingContextId, [contextName])`

Table 2–56 deleteDomainRule Arguments

Argument	Definition
srcDomain	Source domain.
destDomain	Destination domain.
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.23.3 Example

```
deleteDomainRule('dc=oracle,dc=com','dc=oracle,dc=com')
```

2.1.5.24 deleteMappingContext

Delete a mapping context.

2.1.5.24.1 Description Delete the specified mapping context for the libOVD configuration associated with an OPSS context.

2.1.5.24.2 Syntax `deleteMappingContext(mappingContextId, [contextName])`

Table 2–57 deleteMappingContext Arguments

Argument	Definition
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.24.3 Example

```
deleteMappingContext('defaultContext','context)
```

2.1.5.25 deleteView

Deletes a view.

2.1.5.25.1 Description Deletes a view for the libOVD configuration associated with an OPSS context.

2.1.5.25.2 Syntax `createView(viewName, [contextName])`

Table 2–58 createView Arguments

Argument	Definition
viewName	Name of the view to delete.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.25.3 Example

```
deleteView('userView','default')
```

2.1.5.26 getAdapterDetails

Displays the details of an existing adapter.

2.1.5.26.1 Description Displays the details of an existing adapter configured for the libOVD configuration associated with an OPSS context.

2.1.5.26.2 Syntax `getAdapterDetails(adapterName, [contextName])`

Table 2–59 getAdapterDetails Arguments

Argument	Definition
adapterName	Name of the adapter that contains the details to be displayed.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.26.3 Examples

```
getAdapterDetails(adapterName='ldap1', contextName='default')

getAdapterDetails(adapterName='join1')
```

2.1.5.27 listAdapters

Lists the name and type of all adapters.

2.1.5.27.1 Description Lists the name and type of all adapters that are configured for the libOVD configuration associated with an OPSS context.

2.1.5.27.2 Syntax `listAdapters([contextName])`

Table 2–60 listAdapters Arguments

Argument	Definition
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.27.3 Examples

```
listAdapters()

listAdapters(contextName='myContext')
```

2.1.5.28 listAllMappingContextIds

Lists all mapping contexts.

2.1.5.28.1 Description Lists the mapping contexts associated with the specified OPSS context.

2.1.5.28.2 Syntax `listAllMappingContextIds([contextName])`

Table 2–61 listAllMappingContextIds Arguments

Argument	Definition
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.28.3 Example

```
listAllMappingContextIds('default')
```

2.1.5.29 listAttributeRules

Lists all the attribute rules.

2.1.5.29.1 Description List all the attribute rules in the format *SOURCE_ATTRIBUTE*:*DESTINATION_ATTRIBUTE*:*DIRECTION*.

2.1.5.29.2 Syntax `listAttributeRules(mappingContextId, [contextName])`

Table 2–62 listAttributeRules Arguments

Argument	Definition
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.29.3 Example

```
listAttributeRules('defaultContext','default)
```

2.1.5.30 listDomainRules

Lists all domain rules.

2.1.5.30.1 Description Lists all the domain rules in the format of *SOURCE_DOMAIN*:*DESTINATION_DOMAIN*.

2.1.5.30.2 Syntax `listDomainRules(mappingContextId, [contextName])`

Table 2–63 listDomainRules Arguments

Argument	Definition
mappingContextId	Name of the mapping context.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.30.3 Example

```
listDomainRules('defaultContext','default')
```

2.1.5.31 listViews

Lists all views

2.1.5.31.1 Description Lists all views for a libOVD configuration associated with an OPSS context.

2.1.5.31.2 Syntax `listViews([contextName])`

Table 2–64 listViews Arguments

Argument	Definition
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.31.3 Example

```
listViews('default')
```

2.1.5.32 modifyLDAPAdapter

Modifies parameters in an LDAP adapter.

2.1.5.32.1 Description Modifies the following parameters defined in an existing LDAP adapter:

- Remote Base
- Root
- Secure
- BindDN
- BindPassword
- PassCredentials
- MaxPoolSize

2.1.5.32.2 Syntax `modifyLDAPAdapter(adapterName, attribute, value, [contextName])`

Table 2–65 modifyLDAPAdapter Arguments

Argument	Definition
adapterName	Name of the LDAP adapter to be modified.
attribute	Name of the attribute to be modified.
value	New value for the attribute.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.32.3 Examples

```
modifyLDAPAdapter(adapterName='ldap1', attribute='Root', value='dc=us, dc=oracle,
dc=com', contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='RemoteBase', value='dc=org',
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='PassCredentials',
value='BindOnly', contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='BindDN',
value='cn=proxyuser,dc=com', contextName='mydefault')
```

```
modifyLDAPAdapter(adapterName='ldap1', attribute='BindPassword',
value='testwelcome123', contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='Secure', value=true,
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='MaxPoolSize', value=500,
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='MaxPoolChecks', value=10,
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='MaxPoolWait', value=120000,
contextName='mydefault') [value is in milliseconds]

modifyLDAPAdapter(adapterName='ldap1', attribute='InitialPoolSize', value=10,
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='PoolCleanupInterval', value=300,
contextName='mydefault') [value is in seconds]

modifyLDAPAdapter(adapterName='ldap1', attribute='MaxPoolConnectionIdleTime',
value=300, contextName='mydefault')  [value is in seconds]

modifyLDAPAdapter(adapterName='ldap1', attribute='Active', value=false,
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='PingProtocol', value='LDAP',
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='PingBindDN',
value='cn=proxyuser', contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='PingBindPassword',
value='welcome1', contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='PageSize', value=500,
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='HeartBeatInterval', value=120,
contextName='mydefault') [value is in seconds]

modifyLDAPAdapter(adapterName='ldap1', attribute='OperationTimeout', value=120000,
contextName='mydefault') [value is in milliseconds]

modifyLDAPAdapter(adapterName='ldap1', attribute='SearchCountLimit', value=100,
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='Visible', value='Yes',
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='Critical', value='false',
contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='InclusionFilter',
value='objectclass=inetorgperson#base', contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1', attribute='ExclusionFilter',
value='uniquemember=*#base', contextName='mydefault')
```

```
modifyLDAPAdapter(adapterName='ldap1', attribute='DNPattern',
value='(.*)cn=[a-z0-9]*$', contextName='mydefault')

modifyLDAPAdapter(adapterName='ldap1',
attribute='RequestControlAllowServerSupported', value=false,
contextName='mydefault')
```

2.1.5.33 modifySocketOptions

Modifies socket options.

2.1.5.33.1 Description Modifies socket options for an existing LDAP adapter configuration.

2.1.5.33.2 Syntax `modifySocketOptions(adapterName, reuseAddress, keepAlive, tcpNoDelay, readTimeout, [contextName])`

Table 2–66 modifySocketOptions Arguments

Argument	Definition
adapterName	Name of the LDAP adapter to be modified.
reuseAddress	Value of *reuseAddress*.
keepAlive	Value of *keepAlive*.
tcpNoDelay	Value of *tcpNoDelay*.
readTimeout	Value of *readTimeout* in seconds.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.33.3 Example

```
modifySocketOptions(adapterName='ldap1', reuseAddress=true, keepAlive=true,
tcpNoDelay=true, readTimeout=180000, contextName='default')
```

2.1.5.34 removeAllRequestControlExcludeList

Removes all controls from the Request Control Exclude List.

2.1.5.34.1 Description Removes all controls from the Request Control Exclude List for an existing LDAP adapter configuration.

2.1.5.34.2 Syntax `removeAllRequestControlExcludeList(adapterName, [contextName])`

Table 2–67 removeAllRequestControlExcludeList Arguments

Argument	Definition
adapterName	Name of the adapter to be updated.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.34.3 Example

```
removeAllRequestControlExcludeList(adapterName='ldap1', contextName='default')
```

2.1.5.35 removeAllRequestControlIncludeList

Removes all controls from the Request Control Include List.

2.1.5.35.1 Description Removes all controls from the Request Control Include List for an existing LDAP adapter configuration.

2.1.5.35.2 Syntax `removeAllRequestControlIncludeList(adapterName, [contextName])`

Table 2–68 removeAllRequestControlIncludeList Arguments

Argument	Definition
adapterName	Name of the adapter to be updated.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.35.3 Example

```
removeAllRequestControlIncludeList(adapterName='ldap1', contextName='default')
```

2.1.5.36 removeFromRequestControlExcludeList

Removes a control from the Request Control Exclude List.

2.1.5.36.1 Description Removes a control from the Request Control Exclude List for an existing LDAP adapter configuration.

2.1.5.36.2 Syntax `removeFromRequestControlExcludeList(adapterName, control, [contextName])`

Table 2–69 removeFromRequestControlExcludeList Arguments

Argument	Definition
adapterName	Name of the LDAP adapter to be modified.
control	LDAP control object identifier (OID).
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.36.3 Example

```
removeFromRequestControlExcludeList(adapterName='ldap1',
control='2.16.840.1.113894.1.8.31', contextName='default')
```

2.1.5.37 removeDNAttribute

Removes a attribute from the DN Attributes List.

2.1.5.37.1 Description Removes a attribute from the DN Attributes List for an existing adapter that is configured for the libOVD associated with an OPSS context.

2.1.5.37.2 Syntax `removeDNAttribute(adapterName attributeName, [contextName])`

Table 2–70 removeDNAttribute Arguments

Argument	Definition
adapterName	Name of the adapter to be updated.
attributeName	Name of the new DN attribute to be removed.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.37.3 Example

```
removeDNAttribute(adapterName='ldap1', attributeName='memberof',
contextName='default')
```

2.1.5.38 removeFromRequestControlIncludeList

Removes a control from the Request Control Include List.

2.1.5.38.1 Description Removes a control from the Request Control Include List for an existing LDAP adapter configuration.

2.1.5.38.2 Syntax `removeFromRequestControlIncludeList(adapterName, control, [contextName])`

Table 2–71 removeFromRequestControlIncludeList Arguments

Argument	Definition
adapterName	Name of the LDAP adapter to be modified.
control	LDAP control object identifier (OID).
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.38.3 Example

```
removeFromRequestControlIncludeList(adapterName='ldap1',
control='2.16.840.1.113894.1.8.31', contextName='default')
```

2.1.5.39 removeJoinRule

Removes a join rule from a Join Adapter.

2.1.5.39.1 Description Removes a join rule from a Join Adapter configured for the libOVD configuration associated with the specified OPSS context.

2.1.5.39.2 Syntax `removeJoinRule(adapterName, secondary, [contextName])`

Table 2–72 removeJoinRule Arguments

Argument	Definition
adapterName	Name of the Join Adapter to be modified.

Table 2–72 (Cont.) removeJoinRule Arguments

Argument	Definition
secondary	The join rules corresponding to this secondary adapter are removed from the Join Adapter.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.39.3 Examples

```
removeJoinRule('join1','secondaryldap1', 'default')

removeJoinRule(adapterName='join1', secondary='LDAP3')
```

2.1.5.40 removeLDAPHost

Removes a remote host from an existing LDAP adapter.

2.1.5.40.1 Description Removes a remote host (host:port) from an existing LDAP adapter.

2.1.5.40.2 Syntax `removeLDAPHost(adapterName, host, [contextName])`

Table 2–73 removeLDAPHost Arguments

Argument	Definition
adapterName	Name of the LDAP adapter to be modified.
host	Location of a remote LDAP host with which the LDAP adapter will communicate.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.40.3 Examples

```
removeLDAPHost(adapterName='ldap1', host='myhost.example.com')

removeLDAPHost('ldap1', 'myhost.example.com', 'myContext')
```

2.1.5.41 removePlugin

Removes a plug-in from an existing adapter.

2.1.5.41.1 Description Removes a plug-in from an existing adapter or at the global level.

2.1.5.41.2 Syntax `removePlugin(pluginName, [adapterName], [contextName])`

Table 2–74 removePlugin Arguments

Argument	Definition
pluginName	Name of the plug-in to be removed.
adapterName	Optional. Name of the adapter to be modified. If not specified, the global plug-in is removed.

Table 2–74 (Cont.) removePlugin Arguments

Argument	Definition
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.41.3 Examples

```
removePlugin(adapterName='ldap1', pluginName='VirtualAttr')

removePlugin(pluginName='VirtualAttr')
```

2.1.5.42 removePluginParam

Removes an existing parameter from a configured adapter level plug-in.

2.1.5.42.1 Description Removes an existing parameter from a configured adapter level plug-in or a global plug-in. This command removes all values of the particular parameter from the plug-in.

2.1.5.42.2 Syntax `removePluginParam(pluginName, paramKey, [adapterName], [contextName])`

Table 2–75 removePluginParam Arguments

Argument	Definition
pluginName	Name of the plug-in to be modified.
paramKey	Parameter to be removed.
adapterName	Optional. Name of the adapter to be modified. If not specified, the global plug-in is modified.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.42.3 Example

```
removePluginParam(adapterName='ldap1', pluginName='VirtualAttr',
paramKey='ReplaceAttribute')

removePluginParam(pluginName='VirtualAttr', paramKey='ReplaceAttribute')
```

2.1.5.43 replacePluginParam

Replaces existing parameter values for a plug-in.

2.1.5.43.1 Description Replaces existing parameter values for the specified adapter level plug-in or global plug-in.

2.1.5.43.2 Syntax `replacePluginParam(pluginName, paramName, paramValues, [adapterName,][contextName])`

Table 2–76 replacePluginParam Arguments

Argument	Description
pluginName	Name of the plug-in to be modified.
paramName	Name of the parameter to be replaced.
paramValues	New values of the parameter. For more than one new value, separate each new parameter value are by a "|".
adapterName	Optional. Name of the adapter to be modified. If not specified, the global plug-in is modified.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.43.3 Examples

```
replacePluginParam(adapterName='ldap1', pluginName='VirtualAttr',
paramName='ReplaceAttribute', paramValues='cn=%uid%')

replacePluginParam(adapterName='ldap1', pluginName='UserManagement',
paramName='mapAttribute', paramValues='orclguid=objectGuid | uniquemember=member')
```

2.1.5.44 unassignViewFromAdapter

Unassigns a view from an adapter.

2.1.5.44.1 Description Unassigns a view from an LDAP adapter configuration.

2.1.5.44.2 Syntax `unassignViewFromAdapter(viewName, adapterName, [contextName])`

Table 2–77 unassignViewFromAdapter Arguments

Argument	Definition
viewName	Name of the view.
adapterName	Name of the LDAP adapter.
contextName	Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is "default".

2.1.5.44.3 Example

```
unassignViewFromAdapter('userView','ldap1', 'default')
```

3

SSL Configuration WLST Commands

This chapter describes SSL configuration WLST commands.

This chapter contains the following sections:

- About SSL Configuration Commands
- Properties Files for SSL
- configureSSL
- getSSL

3.1 About SSL Configuration Commands

WLST commands are available to configure and manage SSL for Oracle Fusion Middleware components.

Use the commands listed in Table 3–1 for this task.

> **See Also:** *Administering Oracle Fusion Middleware* for important instructions on how to launch the WLST shell to run SSL-related commands. Do not launch the WLST interface from any other location.

> **Note:** All WLST commands for SSL configuration must be run in online mode.

You can obtain help for each command by issuing:

```
help('command_name')
```

Certain commands require parameters like instance name, ias-component and process type. You can obtain this information with the command:

```
state('serverName') [in WebLogic domain]

nmServerStatus(serverName='name', serverType='type') [in Standalone domain]
```

Table 3–1 WLST Commands for SSL Configuration

Use this command...	To...	Use with WLST...
configureSSL	Set the SSL attributes for a component listener.	Online

Table 3–1 (Cont.) WLST Commands for SSL Configuration

Use this command...	To...	Use with WLST...
getSSL	Display the SSL attributes for a component listener.	Online

3.2 Properties Files for SSL

SSL configuration employs certain properties files for use with the WLST `configureSSL` command. The files contain parameters to specify the desired SSL configuration, such as authentication type, cipher values, and SSL version.

You can use descriptive names if you need to manage multiple properties files for different components. For example, you could have properties files named `ohs-ssl-properties.prop` or `ovd-ssl-properties.prop`.

3.2.1 Structure of Properties Files

All the SSL properties files have a consistent structure.

Table 3–2 provides details about the key-value structure and usage of these files.

Table 3–2 Parameters in Properties File

Key	Mandatory?	Allowed Values for Oracle HTTP Server	Usage
SSLEnabled	No	true false	Either value
Ciphers	No	See "SSLCipherSuite" in *Administering Oracle HTTP Server*.	One or more comma separated values
SSLVersions	No	See "SSLProtocol" in *Administering Oracle HTTP Server*.	
CertValidation	No	none crl	Either value
CertValidation Path	No	file://`crl_file_path` dir://`crl_dir_path`	Path of the CRL file, or directory containing CRL files
KeyStore	No	Valid wallet name	
TrustStore	No	N/A	
AuthenticationType	No	None Server Optional Mutual	Any one value

Table 3–3 shows the default values:

Table 3–3 Default Values of Parameters

Key	Default Value for Oracle HTTP Server
SSLEnabled	true
Ciphers	null
SSLVersions	null
CertValidation	none
CertValidation Path	null

Table 3–3 (Cont.) Default Values of Parameters

Key	Default Value for Oracle HTTP Server
KeyStore	default
TrustStore	-
Authentication Type	Server

Note:

- At least one `DH_anon` cipher must be used in SSL no-auth mode. For all other modes, at least one `RSA` cipher must be used.
- The value of the `KeyStore` parameter must be specified when configuring SSL in server-auth, mutual-auth, or optional client auth.
- If only `AES` ciphers have been specified, the `SSLVersions` parameter must contain `TLSv1` or `nzos_Version_1_0`.
- If you are doing CRL-based validation, the value of the `CertValidation` parameter should be `crl` and the value of the `CertValidationPath` parameter should point to the CRL file/directory.

3.2.2 Examples of Properties Files

Some examples demonstrating the use of the properties files follow.

Example 1: Basic Properties File

```
SSLEnabled=true
AuthenticationType=None
CertValidation=none
```

This properties file specifies no authentication mode, and default values will be used during SSL configuration for ciphers and SSL version. Keystore and truststore properties are not specified since the authentication type is `None`. For other authentication types, keystore must be specified.

Example 2: Basic Properties File

```
SSLEnabled=
AuthenticationType=None
CertValidation=none
```

This properties file is exactly the same as above, except that `SSLEnabled` is explicitly specified without any value. This is the same as not specifying the key at all. In both cases, the default value will be used.

Therefore, all the following three settings have the same meaning:

- The setting:

  ```
  SSLEnabled=true
  ```

  Here the value `true` is explicitly specified.

- The setting:

```
SSLEnabled=
```

Since no value is mentioned here, the default value of `SSLEnabled` (`true`) is used.

- The key `SSLEnabled` is not present in the properties file.

  Since the key is not present, its default value (`true`) is used.

Example 3: Properties File with Version for OHS

```
SSLEnabled=true
AuthenticationType=Mutual
SSLVersion=nzos_Version_1_0
CertValidation=crl
CertValidationPath=file:///tmp/file.crl
KeyStore=ohs1
```

This properties file has:

- Default values for ciphers
- Keystore
- SSL version v1
- CRL validation turned on
- Mutual Authentication mode

3.3 configureSSL

Online command that sets SSL attributes.

3.3.1 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, then default attribute values are used.

For details about the format of properties files, see Section 3.2, "Properties Files for SSL."

3.3.2 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 value is 'ohs'.
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.

3.3.3 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')
```

3.4 getSSL

Online command that lists the configured SSL attributes.

3.4.1 Description

This command lists the configured SSL attributes for the specified component listener.

3.4.2 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 value is 'ohs'.
listener	Specifies the name of the component listener.

3.4.3 Example

The following command shows the SSL attributes configured for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`, for listener `sslport1`:

```
wls:/mydomain/serverConfig> getSSL('inst1', 'ohs1', 'ohs', 'sslport1')
```

4

Wallet Configuration WLST Commands

This chapter describes how to configure Oracle wallets.

This chapter contains these topics:

4.1 Background for WLST Wallet Commands

WLST commands are available to manage Oracle wallets for Oracle Fusion Middleware components.

Use the commands listed in Table 4–1 for this task.

> **See Also:** *Administering Oracle Fusion Middleware* for important instructions on how to launch the WLST shell to run SSL-related commands. Do not launch the WLST interface from any other location.

> **Note:** All WLST commands for SSL configuration must be run in online mode.

You can obtain help for each command by issuing:

```
help('command_name')
```

Certain commands require parameters like instance name, ias-component and process type. You can obtain this information with the command:

```
state('serverName') [in WebLogic domain]

nmServerStatus(serverName='name', serverType='type') [in Standalone domain]
```

Table 4–1 WLST Commands for Oracle Wallet Management

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
changeWalletPassword	Change the password to an Oracle wallet.	Online
createWallet	Create an Oracle wallet.	Online
deleteWallet	Delete an Oracle wallet.	Online

Table 4–1 (Cont.) WLST Commands for Oracle Wallet Management

Use this command...	To...	Use with WLST...
exportWallet	Export an Oracle wallet to a file.	Online
exportWalletObject	Export an object (for example, a certificate) from an Oracle wallet to a file.	Online
getWalletObject	Display a certificate or other object present in an Oracle wallet.	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
listWalletObjects	List all objects (such as certificates) present in an Oracle wallet.	Online
listWallets	List all Oracle wallets configured for a component instance.	Online
removeWalletObject	Remove a certificate or other object from a component instance's Oracle wallet.	Online

4.2 WLST Reference for Wallet Management

Note: `WLST` allows you to import certificates only in PEM format.

4.2.1 addCertificateRequest

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

4.2.1.1 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). To get a certificate signed by a certificate authority (CA), send the certificate signing request to your CA.

4.2.1.2 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 value is 'ohs'.
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.

4.2.1.3 Example

The following command generates a certificate signing request with DN `cn=www.acme.com` and key size `1024` in `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> addCertificateRequest('inst1', 'ohs1',
'ohs','wallet1', 'password', 'cn=www.acme.com', '1024',)
```

4.2.2 addSelfSignedCertificate

Online command that adds a self-signed certificate.

4.2.2.1 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). Only keys based on the RSA algorithm are generated.

4.2.2.2 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 value is 'ohs'.
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.

4.2.2.3 Example

The following command adds a self-signed certificate with DN `cn=www.acme.com`, key size `1024` to `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance inst1:

```
wls:/mydomain/serverConfig> addSelfSignedCertificate('inst1', 'ohs1',
'ohs','wallet1', 'password', 'cn=www.acme.com', '1024')
```

4.2.3 changeWalletPassword

Online command that changes the password of an Oracle wallet.

4.2.3.1 Description

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

4.2.3.2 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 value is 'ohs'.
walletName	Specifies the filename of the wallet.
currPassword	Specifies the current wallet password.
newPassword	Specifies the new wallet password.

4.2.3.3 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')
```

4.2.4 createWallet

Online command that creates an Oracle wallet.

4.2.4.1 Description

This command creates an Oracle wallet for the specified component instance (Oracle HTTP Server). Wallets can be of password-protected or auto-login type.

4.2.4.2 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 value is 'ohs'.
walletName	Specifies the name of the wallet file to be created.
password	Specifies the wallet password.

4.2.4.3 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', '')
```

4.2.5 deleteWallet

Online command that deletes an Oracle wallet.

4.2.5.1 Description

This command deletes an Oracle wallet for the specified component instance.

4.2.5.2 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 value is 'ohs'.
walletName	Specifies the name of the wallet file to be deleted.

4.2.5.3 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')
```

4.2.6 exportWallet

Online command that exports an Oracle wallet.

4.2.6.1 Description

This command exports an Oracle wallet, configured for a specified component instance, to files 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.

4.2.6.2 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 value is 'ohs'.
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.

4.2.6.3 Examples

The following command exports auto-login wallet `wallet1` for Oracle HTTP Server instance `ohs1` to file `cwallet.sso` under `/tmp`:

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

The following command exports password-protected wallet `wallet2` for Oracle HTTP Server instance `ohs1` to two files, `ewallet.p12` and `cwallet.sso`, under `/tmp`:

```
wls:/mydomain/serverConfig> exportWallet('inst1', 'ohs1', 'ohs', 'wallet2',
'password', '/tmp')
```

4.2.7 exportWalletObject

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

4.2.7.1 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. DN indicates the object to be exported.

4.2.7.2 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 value is 'ohs'.
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.

4.2.7.3 Examples

The following command exports a certificate signing request with DN `cn=www.acme.com` in `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`. The certificate signing request is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportWalletObject('inst1', 'ohs1', 'ohs','wallet1',
'password', 'CertificateRequest', '/tmp','cn=www.acme.com')
```

The following command exports a certificate with DN `cn=www.acme.com` in `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`. The certificate or certificate chain is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportWalletObject('inst1', 'ohs1', 'ohs','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 HTTP Server instance `ohs1`, in application server instance `inst1`. The trusted certificate is exported under the directory `/tmp`:

```
wls:/mydomain/serverConfig> exportWalletObject('inst1', 'ohs1', 'ohs','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 HTTP Server instance ohs1, in application server instance inst1. The certificate or certificate chain is exported under the directory /tmp:

```
wls:/mydomain/serverConfig> exportWalletObject('inst1', 'ohs1', 'ohs','wallet1',
'password', 'TrustedChain', '/tmp','cn=www.acme.com')
```

4.2.8 getWalletObject

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

4.2.8.1 Description

This command displays a specific certificate signing request, certificate or trusted certificate present in an Oracle wallet for the specified component instance. 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.

4.2.8.2 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 value is 'ohs'.
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.

4.2.8.3 Examples

The following command shows certificate signing request details for the object with index 0 present in wallet1, for Oracle HTTP Server instance ohs1, in application server instance inst1:

```
wls:/mydomain/serverConfig> getKeyStoreObject('inst1', 'ohs1',
'ohs','wallet1','password', 'CertificateRequest', '0')
```

The following command shows certificate details for the object with index 0 present in wallet1, for Oracle HTTP Server instance ohs1, in application server instance inst1:

```
wls:/mydomain/serverConfig> getKeyStoreObject('inst1', 'ohs1',
'ohs','wallet1','password', 'Certificate', '0')
```

The following command shows trusted certificate details for the object with index 0, present in wallet1, for Oracle HTTP Server instance ohs1, in application serverinstance inst1:

```
wls:/mydomain/serverConfig> getKeyStoreObject('inst1', 'ohs1',
'ohs','wallet1','password', 'TrustedCertificate', '0')
```

4.2.9 importWallet

Online command that imports an Oracle wallet from a file.

4.2.9.1 Description

This command imports an Oracle wallet from a file to the specified component instance 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.

4.2.9.2 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 value is 'ohs'.
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.

4.2.9.3 Examples

The following command imports the auto-login wallet file `/tmp/cwallet.sso` as `wallet1` into Oracle HTTP Server instance `ohs1`. 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', 'ohs1', 'ohs', 'wallet1', '',
'/tmp/cwallet.sso')
```

The following command imports password-protected wallet `/tmp/ewallet.p12` as `wallet2` into Oracle HTTP Server instance `ohs1`. Subsequently, the wallet is managed with the name `wallet2`. The wallet password is passed as a parameter:

```
wls:/mydomain/serverConfig> importWallet('inst1', 'ohs1', 'ohs', 'wallet2',
'password', '/tmp/ewallet.p12')
```

4.2.10 importWalletObject

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

4.2.10.1 Description

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

4.2.10.2 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 value is 'ohs'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
type	Specifies the type of wallet object to be imported. Valid values are 'Certificate', 'TrustedCertificate' and 'TrustedChain'.
filePath	Specifies the absolute path of the file containing the wallet object.

4.2.10.3 Examples

The following command imports a certificate chain in PKCS#7 format from file `chain.txt` into `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> importWalletObject('inst1', 'ohs1', 'ohs','wallet1',
'password', 'TrustedChain','/tmp/chain.txt')
```

The following command imports a certificate from file `cert.txt` into `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> > importWalletObject('inst1', 'ohs1', 'ohs','wallet1',
'password', 'Certificate','/tmp/cert.txt')
```

The following command imports a trusted certificate from file `trust.txt` into `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> importWalletObject('inst1', 'ohs1', 'ohs','wallet1',
'password', 'TrustedCertificate','/tmp/trust.txt')
```

4.2.11 listWalletObjects

Online command that lists all objects in an Oracle wallet.

4.2.11.1 Description

This command lists all certificate signing requests, certificates, or trusted certificates present in an Oracle wallet for the specified component instance.

4.2.11.2 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 value is 'ohs'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.

Argument	Definition
type	Specifies the type of wallet object to be listed. Valid values are 'CertificateRequest', 'Certificate', and 'TrustedCertificate'.

4.2.11.3 Examples

The following command lists all certificate signing requests in `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`:

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

The following command lists all certificates in `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`:

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

The following command lists all trusted certificates in `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`:

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

4.2.12 listWallets

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

4.2.12.1 Description

This command displays all the wallets configured for the specified component instance, and identifies the auto-login wallets.

4.2.12.2 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 value is 'ohs'.

4.2.12.3 Example

The following command lists all wallets for Oracle HTTP Server instance `ohs1` in application server instance `inst1`:

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

4.2.13 removeWalletObject

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

4.2.13.1 Description

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

4.2.13.2 Syntax

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

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

4.2.13.3 Examples

The following command removes all trusted certificates from `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`. It is not necessary to provide a DN, so you pass null (denoted by `None`) for the DN parameter:

```
wls:/mydomain/serverConfig> removeWalletObject('inst1', 'ohs1', 'ohs','wallet1',
'password', 'TrustedAll',None)
```

The following command removes a certificate signing request indicated by DN `cn=www.acme.com` from `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> removeWalletObject('inst1', 'ohs1', 'ohs','wallet1',
'password', 'CertificateRequest','cn=www.acme.com')
```

The following command removes a certificate indicated by DN `cn=www.acme.com` from `wallet1`, for Oracle HTTP Server instance `ohs1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> removeWalletObject('inst1', 'ohs1', 'ohs','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 HTTP Server instance `ohs1`, in application server instance `inst1`:

```
wls:/mydomain/serverConfig> removeWalletObject('inst1', 'ohs1', 'ohs','wallet1',
'password', 'TrustedCertificate','cn=www.acme.com')
```