This document describes the Oracle Fusion Middleware Infrastructure Security commands available to use with the WebLogic Scripting Tool.
# 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 ............................................................................................................................ 2-24  
2.1.3 OPSS Keystore Service Commands .......................................................................................................................... 2-34  
2.1.4 Identity Directory Service Commands .................................................................................................................... 2-47  
2.1.5 Library Oracle Virtual Directory (libOVD) Commands ............................................................................................ 2-68  

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  

4 **Wallet Configuration WLST Commands**

4.1 The WLST Wallet Commands .......................................................................................................................................... 4-1
Preface

This guide describes the security WebLogic Scripting Tool (WLST) commands for the 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 that have purchased support 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/technetwork/indexes/documentation/index.html.

Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface</td>
<td>Boldface type indicates graphical user interface elements associated with an action.</td>
</tr>
<tr>
<td>Convention</td>
<td>Meaning</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, terms defined in text, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><em>monospace</em></td>
<td>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.</td>
</tr>
</tbody>
</table>
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, Oracle Identify Federation, Directory Integration Platform, OPSS, Oracle Keystore Service, Identity Directory Service, and Library Oracle Virtual Directory (libOVD).
- **Chapter 3, "SSL Configuration WLST Commands"** provides detailed descriptions for the SSL configuration WLST commands.
- **Chapter 4, "Wallet Configuration WLST Commands"** provides detailed descriptions of the WLST commands that you can use to configure Oracle wallets.
This chapter describes the Oracle Fusion Middleware Infrastructure Security WLST commands.

It includes the following topic:

- **Infrastructure Commands**

**Related Topics**

*Securing Applications with Oracle Platform Security Services.*

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

**Note:** In syntax descriptions, optional arguments are enclosed in between square brackets; all other arguments are required.

### 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

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

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

**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.

**Syntax**

```
addBootStrapCredential(jpsConfigFile, map, key, username, password)
```

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpsConfigFile</td>
<td>Specifies the location of the file jps-config.xml relative to the location where the command is run.</td>
</tr>
<tr>
<td>map</td>
<td>Specifies the map of the credential to add.</td>
</tr>
<tr>
<td>key</td>
<td>Specifies the key of the credential to add.</td>
</tr>
<tr>
<td>username</td>
<td>Specifies the name of the user in the credential to add.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the user in the credential to add.</td>
</tr>
</tbody>
</table>

**Example**

The following example 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.
**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.

**Syntax**
```
addResourceToEntitlement(appStripe="appStripeName", name="entName", resourceName="resName", actions="actionList")
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the entitlement is located.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the entitlement to modify.</td>
</tr>
<tr>
<td>resourceName</td>
<td>Specifies the name of the resource to add.</td>
</tr>
<tr>
<td>resourceType</td>
<td>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.</td>
</tr>
<tr>
<td>actions</td>
<td>Specifies the comma-separated list of actions for the added resource.</td>
</tr>
</tbody>
</table>

**Example**
The following example 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.

**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.

**Syntax**
```
createAppRole(appStripe, appRoleName)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies an application stripe.</td>
</tr>
<tr>
<td>appRoleName</td>
<td>Specifies a role name.</td>
</tr>
</tbody>
</table>

**Example**
The following example 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.
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.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>Specifies a map name (folder).</td>
</tr>
<tr>
<td>key</td>
<td>Specifies a key name.</td>
</tr>
<tr>
<td>user</td>
<td>Specifies the credential user name.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the credential password.</td>
</tr>
<tr>
<td>desc</td>
<td>Specifies a string describing the credential.</td>
</tr>
</tbody>
</table>

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

```
wlsh:/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.

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.

Syntax
createEntitlement(appStripe="appStripeName", name="entitlementName", resourceName="resName", actions="actionList" [,-displayName="dispName"] [,-description="descript"])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the entitlement is created.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the entitlement created.</td>
</tr>
<tr>
<td>resourceName</td>
<td>Specifies the name of the one resource member of the entitlement created.</td>
</tr>
<tr>
<td>actions</td>
<td>Specifies a comma-separated the list of actions for the resource resourceName.</td>
</tr>
<tr>
<td>displayName</td>
<td>Specifies the display name of the resource created. Optional.</td>
</tr>
<tr>
<td>description</td>
<td>Specifies the description of the entitlement created. Optional.</td>
</tr>
</tbody>
</table>

Example
The following example creates the entitlement myEntitlement with just the resource myResource in the stripe myApplication:
2.1.1.6 createResource
Online command that creates a new resource.

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

Syntax
createResource(appStripe="appStripeName", name="resName", type="resTypeName" [,-displayName="dispName"] [,-description="descript"])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the resource is created.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the resource created.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of resource created. The passed resource type must be present in the application stripe at the time this script is invoked.</td>
</tr>
<tr>
<td>displayName</td>
<td>Specifies the display name of the resource created. Optional.</td>
</tr>
<tr>
<td>description</td>
<td>Specifies the description of the resource created. Optional.</td>
</tr>
</tbody>
</table>

Example
The following example creates the resource myResource in the stripe myApplication:

```
wlshostDomain/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.

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

Syntax
createResourceType(appStripe, resourceTypeName, displayName, description [,[ provider] [,[ matcher]], actions [,[ delimiter]])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where to insert the resource type.</td>
</tr>
<tr>
<td>resourceTypeName</td>
<td>Specifies the name of the resource type to insert.</td>
</tr>
<tr>
<td>displayName</td>
<td>Specifies the name for the resource type used in UI gadgets.</td>
</tr>
<tr>
<td>description</td>
<td>Specifies a brief description of the resource type.</td>
</tr>
<tr>
<td>provider</td>
<td>Specifies the provider for the resource type.</td>
</tr>
<tr>
<td>matcher</td>
<td>Specifies the matcher for the resource type.</td>
</tr>
<tr>
<td>delimiter</td>
<td>Specifies the delimiter for the resource type.</td>
</tr>
</tbody>
</table>
Infrastructure Commands

Infrastructure Security Custom WLST Commands

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

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

Syntax
deleteAppPolicies(appStripe)

Example
The following example removes all policies of application myApp:

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

2.1.9 deleteAppRole
Online command that removes an application role.

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.

Syntax
createAppRole(appStripe, appRoleName)

Example
The following example removes an application role:

wls:/mydomain/serverConfig> createAppRole(appStripe="myApplication", appRoleName="roleName")
Example
The following example removes the role with application stripe myApp and role name myRole:

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

2.1.10 deleteEntitlement
Online command that deletes an entitlement.

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.

Syntax
```
deleteEntitlement(appStripe="appStripeName", name="entitlementName")
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the entitlement is deleted.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the entitlement to delete.</td>
</tr>
</tbody>
</table>

Example
The following example deletes the entitlement myEntitlement in the stripe myApplication:

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

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

Description
Removed 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.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>Specifies a map name (folder).</td>
</tr>
<tr>
<td>key</td>
<td>Specifies a key name.</td>
</tr>
</tbody>
</table>

Example
The following example removes the credential with map name myMap and key name myKey:

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

2.1.12 deleteResource
Online command that deletes a resource.
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.

Syntax
`deleteResource(appStripe="appStripeName", name="resName", type="resTypeName")`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the resource is deleted.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the resource deleted.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of resource deleted. The passed resource type must be present in the application stripe at the time this script is invoked.</td>
</tr>
</tbody>
</table>

Example
The following example deletes the resource myResource in the stripe myApplication:

```
wlst:/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.

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`.

Syntax
`deleteResourceType(appStripe, resourceTypeName)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe from where to remove the resource type.</td>
</tr>
<tr>
<td>resourceTypeName</td>
<td>Specifies the name of the resource type to remove.</td>
</tr>
</tbody>
</table>

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

```
wlst:/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.
**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`.

```
exportEncryptionKey(jpsConfigFile, keyFilePath, keyFilePassword)
```

**Syntax**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>jpsConfigFile</strong></td>
<td>Specifies the location of the file <code>jps-config.xml</code> relative to the location where the command is run.</td>
</tr>
<tr>
<td><strong>keyFilePath</strong></td>
<td>Specifies the directory where the file <em>ewallet.p12</em> is created; note that the content of this file is encrypted and secured by the value passed to <code>keyFilePassword</code>.</td>
</tr>
<tr>
<td><strong>keyFilePassword</strong></td>
<td>Specifies the password to secure the file <em>ewallet.p12</em>; note that this same password must be used when importing that file.</td>
</tr>
</tbody>
</table>

**Example**
The following example 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.

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

**Syntax**

```
getEntitlement(appStripe="appStripeName", name="entitlementName")
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>appStripe</strong></td>
<td>Specifies the application stripe where the entitlement is located.</td>
</tr>
<tr>
<td><strong>name</strong></td>
<td>Specifies the name of the entitlement to access.</td>
</tr>
</tbody>
</table>

**Example**
The following example 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.
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.

Syntax
getResourceType(appStripe, resourceTypeName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe from where to fetch the resource type.</td>
</tr>
<tr>
<td>resourceTypeName</td>
<td>Specifies the name of the resource type to fetch.</td>
</tr>
</tbody>
</table>

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

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

2.1.17 grantAppRole
Online command that adds a principal to a role.

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.

Syntax
grantAppRole(appStripe, appRoleName,principalClass, principalName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies an application stripe.</td>
</tr>
<tr>
<td>appRoleName</td>
<td>Specifies a role name.</td>
</tr>
<tr>
<td>principalClass</td>
<td>Specifies the fully qualified name of a class.</td>
</tr>
<tr>
<td>principalName</td>
<td>Specifies the principal name.</td>
</tr>
</tbody>
</table>

Example
The following example 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")

2.1.18 grantEntitlement
Online command that grant an entitlement to a named principal.

Description
Grants an entitlement to a specified principal in a specified application stripe.
Syntax
grantEntitlement(appStripe="appStripeName", principalClass="principalClass", principalName="principalName", permSetName="entName")

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the principal resides.</td>
</tr>
<tr>
<td>principalClass</td>
<td>Specifies the class associated with the principal.</td>
</tr>
<tr>
<td>principalName</td>
<td>Specifies the name of the principal to which the entitlement is granted.</td>
</tr>
<tr>
<td>permSetName</td>
<td>Specifies the name of the entitlement granted.</td>
</tr>
</tbody>
</table>

Example
The following example grants the entitlement myEntitlement in the stripe myApplication to the principal myPrincipalName:

```
wlst:/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.

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

Syntax
grantPermission([appStripe,] [codeBaseURL,] [principalClass,] [principalName,] permClass, [permTarget,] [permActions])

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

Example
The following example 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 example 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.

**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`.

**Syntax**

```
importEncryptionKey(jpsConfigFile, keyFilePath, keyFilePassword)
```

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpsConfigFile</td>
<td>Specifies the location of the file jps-config.xml relative to the location where the command is run.</td>
</tr>
<tr>
<td>keyFilePath</td>
<td>Specifies the directory where the ewallet.p12 is located.</td>
</tr>
<tr>
<td>keyFilePassword</td>
<td>Specifies the password used when the file ewallet.p12 was generated.</td>
</tr>
</tbody>
</table>

**Example**

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

### 2.1.1.21 listAppRoles

Online command that lists all roles in an application.

**Description**

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

**Syntax**

```
listAppRoles(appStripe)
```

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies an application stripe.</td>
</tr>
</tbody>
</table>

**Example**

The following example 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.
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.

Syntax
listAppRoleMembers(appStripe, appRoleName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies an application stripe.</td>
</tr>
<tr>
<td>appRoleName</td>
<td>Specifies a role name.</td>
</tr>
</tbody>
</table>

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

```
wlst:/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.

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.

Syntax
listAppStripes([configFile="configFileName"] [, regularExpression="aRegExp"])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>configFile</td>
<td>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.</td>
</tr>
<tr>
<td>regularExpression</td>
<td>Specifies the regular expression that returned stripe names should match. Optional. If unspecified, it matches all names. To match substrings, use the character &quot;.&quot;</td>
</tr>
</tbody>
</table>

Example
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:
The following (online) invocation returns the list of application stripes that contain the prefix App:

```
wlst:/mydomain/serverConfig> listAppStripes(RegularExpression="App*")
```

### 2.1.1.24 listCodeSourcePermissions

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

**Description**

This command allows listing codebase permissions in global policies.

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>codeBaseURL</td>
<td>Specifies the name of the grantee codebase URL.</td>
</tr>
</tbody>
</table>

**Example**

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

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

### 2.1.1.25 listEntitlement

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

**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.

**Syntax**

```
listEntitlement(appStripe="appStripeName" [, principalName="principalName", principalClass="principalClass"])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the entitlement is deleted.</td>
</tr>
<tr>
<td>principalName</td>
<td>Specifies the name of the principal to match. Optional.</td>
</tr>
<tr>
<td>principalClass</td>
<td>Specifies the class of the principal to match. Optional.</td>
</tr>
</tbody>
</table>

**Example**

The following example lists all entitlements in the stripe myApplication:

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

### 2.1.1.26 listEntitlements

Online command that lists the entitlements in an application stripe.
**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.

**Syntax**

```
listEntitlements(appStripe="appStripeName", resourceTypeName="resTypeName", resourceName="resName")
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe from where to list entitlements.</td>
</tr>
<tr>
<td>resourceTypeName</td>
<td>Specifies the name of the type of the resources to list. Optional.</td>
</tr>
<tr>
<td>resourceName</td>
<td>Specifies the name of resource to match. Optional.</td>
</tr>
</tbody>
</table>

**Examples**
The following example lists all the entitlements in the stripe myApplication:

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

The following example 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.

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

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies an application stripe. If not specified, the command works on system policies.</td>
</tr>
<tr>
<td>principalClass</td>
<td>Specifies the fully qualified name of a class (grantee).</td>
</tr>
<tr>
<td>principalName</td>
<td>Specifies the name of the grantee principal.</td>
</tr>
</tbody>
</table>

**Example**
The following example 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 example lists all permissions granted to a principal by system policies:
2.1.1.28 listResourceActions

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

**Description**

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

**Syntax**

```
listResourceActions(appStripe="appStripeName", permSetName="entitlementName")
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the entitlement resides.</td>
</tr>
<tr>
<td>permSetName</td>
<td>Specifies the name of the entitlement whose resources and actions to list.</td>
</tr>
</tbody>
</table>

**Example**

The following example lists the resources and actions of the entitlement myEntitlement in the stripe myApplication:

```
wlsh:/mydomain/serverConfig> listResourceActions(appStripe="myApplication", permSetName="myEntitlement")
```

2.1.1.29 listResources

Online command that lists resources in a specified application stripe.

**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.

**Syntax**

```
listResources(appStripe="appStripeName" [,type="resTypeName"])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the resources are listed.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of resource listed. The passed resource type must be present in the application stripe at the time this script is invoked.</td>
</tr>
</tbody>
</table>

**Example**

The following example lists all resources of type myResType in the stripe myApplication:

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

2.1.1.30 listResourceTypes

Online command that lists resource types.
Description
Lists all the resource types in a specified application stripe.

Syntax
listResourceTypes{appStripe="appStripeName"}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the resource types are located.</td>
</tr>
</tbody>
</table>

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

```
wlst:/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.

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).

Syntax
listSecurityStoreInfo{domainConfig="configFilePath"}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainConfig</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

```
wlst:/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.
**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.

**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.

**Syntax**
modifyBootStrapCredential(jpsConfigFile, username, password)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpsConfigFile</td>
<td>Specifies the location of the file jps-config.xml relative to the location where the command is run.</td>
</tr>
<tr>
<td>username</td>
<td>Specifies the distinguished name of the user in the LDAP store.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the user.</td>
</tr>
</tbody>
</table>

**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 example changes the password in the bootstrap credential store to welcome1:

```bash
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.

**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.

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

**Syntax**
restoreEncryptionKey(jpsConfigFile)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpsConfigFile</td>
<td>Specifies the location of the file jps-config.xml relative to the location where the command is run.</td>
</tr>
</tbody>
</table>

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

2.1.1.36 revokeAppRole
Online command that removes a principal from a role.

**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.

**Syntax**
revokeAppRole(appStripe, appRoleName, principalClass, principalName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies an application stripe.</td>
</tr>
<tr>
<td>appRoleName</td>
<td>Specifies a role name.</td>
</tr>
<tr>
<td>principalClass</td>
<td>Specifies the fully qualified name of a class.</td>
</tr>
<tr>
<td>principalName</td>
<td>Specifies the principal name.</td>
</tr>
</tbody>
</table>

**Example**
The following example 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.

**Description**
Deletes an entitlement and revokes the entitlement from the principal in a specified application stripe.
Syntax
revokeEntitlement(appStripe="appStripeName", principalClass="principalClass",
principalName="principalName", permSetName="entName")

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the entitlement is deleted.</td>
</tr>
<tr>
<td>principalClass</td>
<td>Specifies the class associated with the principal.</td>
</tr>
<tr>
<td>principalName</td>
<td>Specifies the name of the principal to which the entitlement is revoked.</td>
</tr>
<tr>
<td>permSetName</td>
<td>Specifies the name of the entitlement deleted.</td>
</tr>
</tbody>
</table>

Example
The following example deleted the entitlement myEntitlement in the stripe myApplication:

```
wlst:/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.

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

Syntax
revokePermission([appStripe,] [codeBaseURL,] [principalClass,] [principalName,]
permClass, [permTarget,] [permActions])

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

Example
The following example removes the application permission (for the application with application stripe myApp) with the specified data:

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

**Description**

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

**Syntax**

```
revokeResourceFromEntitlement(appStripe="appStripeName", name="entName", resourceName="resName", resourceType="resTypeName", actions="actionList")
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>appStripe</td>
<td>Specifies the application stripe where the entitlement is located.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the entitlement to modify.</td>
</tr>
<tr>
<td>resourceName</td>
<td>Specifies the name of the resource to remove.</td>
</tr>
<tr>
<td>resourceType</td>
<td>Specifies the type of the resource to remove.</td>
</tr>
<tr>
<td>actions</td>
<td>Specifies the comma-separated list of actions to remove.</td>
</tr>
</tbody>
</table>

**Example**

The following example 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.

**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`.
Syntax
rollOverEncryptionKey(jpsConfigFile="pathName")

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpsConfigFile</td>
<td>Specifies the location of the file jps-config.xml; either relative to the</td>
</tr>
<tr>
<td></td>
<td>location where the script is run, or the full path.</td>
</tr>
</tbody>
</table>

Example
The following example 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.

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.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>Specifies a map name (folder).</td>
</tr>
<tr>
<td>key</td>
<td>Specifies a key name.</td>
</tr>
<tr>
<td>user</td>
<td>Specifies the credential user name.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the credential password.</td>
</tr>
<tr>
<td>desc</td>
<td>Specifies a string describing the credential.</td>
</tr>
</tbody>
</table>

Example
The following example 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.

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

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:

```plaintext
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.

### Example

The following example updates the trust store service with the specifications in the file `myProps`:

```plaintext
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>
<thead>
<tr>
<th>Use this command</th>
<th>To</th>
<th>Use with WLST</th>
</tr>
</thead>
<tbody>
<tr>
<td>createIAUView</td>
<td>Generate an SQL script to create an IAU view in the database.</td>
<td>Online</td>
</tr>
<tr>
<td>createAuditDBView</td>
<td>Generate an SQL script to create an audit definitions view in the database.</td>
<td>Online</td>
</tr>
<tr>
<td>deregisterAudit</td>
<td>Remove audit definitions of a specified component from the audit store.</td>
<td>Online</td>
</tr>
<tr>
<td>exportAuditConfig</td>
<td>Export a component's audit configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>getIAUViewInfo</td>
<td>Get information about a view.</td>
<td>Online</td>
</tr>
<tr>
<td>getNonJavaEEAuditMBeanName</td>
<td>Display the mBean name for a non-Java EE component.</td>
<td>Online</td>
</tr>
<tr>
<td>getAuditPolicy</td>
<td>Display audit policy settings.</td>
<td>Online</td>
</tr>
<tr>
<td>getAuditRepository</td>
<td>Display audit repository settings.</td>
<td>Online</td>
</tr>
<tr>
<td>importAuditConfig</td>
<td>Import a component's audit configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>listAuditComponents</td>
<td>List components that can be audited.</td>
<td>Online</td>
</tr>
<tr>
<td>listAuditEvents</td>
<td>List audit events for one or all components.</td>
<td>Online</td>
</tr>
<tr>
<td>setAuditPolicy</td>
<td>Update audit policy settings.</td>
<td>Online</td>
</tr>
<tr>
<td>setAuditRepository</td>
<td>Update audit repository settings.</td>
<td>Online</td>
</tr>
</tbody>
</table>
2.1.2.1 `getNonJavaEEAuditMBeanName`

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

**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.

**Syntax**

```java
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.

**Example**

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

```bash
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.

**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.

**Syntax**

```java
getAuditPolicy([mbeanName, componentType])
```
Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbeanName</td>
<td>Specifies the name of the component audit MBean for non-Java EE components.</td>
</tr>
<tr>
<td>componentType</td>
<td>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.</td>
</tr>
</tbody>
</table>

Example

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

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

2.1.2.3 setAuditPolicy

Online command that updates an audit policy.

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.

Syntax

`setAuditPolicy([mbeanName],[filterPreset],[addSpecialUsers],[removeSpecialUsers],[addCustomEvents],[removeCustomEvents],[componentType],[maxFileSize],[andCriteria],[orCriteria],[componentEventsFile])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbeanName</td>
<td>Specifies the name of the component audit MBean for non-Java EE components.</td>
</tr>
<tr>
<td>filterPreset</td>
<td>Specifies the filter preset to be changed.</td>
</tr>
<tr>
<td>addSpecialUsers</td>
<td>Specifies the special users to be added.</td>
</tr>
<tr>
<td>removeSpecialUsers</td>
<td>Specifies the special users to be removed.</td>
</tr>
<tr>
<td>addCustomEvents</td>
<td>Specifies the custom events to be added.</td>
</tr>
<tr>
<td>removeCustomEvents</td>
<td>Specifies the custom events to be removed.</td>
</tr>
</tbody>
</table>
Examples
The following example sets audit policy to None level, and adds users user2 and user3 while removing user1 from the policy:

```
wlst:/mydomain/serverConfig> setAuditPolicy (filterPreset='None',addSpecialUsers='user2,user3',removeSpecialUsers='user1',componentType='JPS')
wlst:/mydomain/serverConfig> getAuditPolicy(componentType='JPS');
Already in Domain Runtime Tree
FilterPreset:None
Special Users:user2,user3
Max Log File Size:104857600
```

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

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

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

The following example 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
wlst:/IDMDomain/domainRuntime> getAuditPolicy(componentType='JPS');
```

---

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>componentType</td>
<td>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.</td>
</tr>
<tr>
<td>maxFileSize</td>
<td>Specifies the maximum size of the log file.</td>
</tr>
<tr>
<td>andCriteria</td>
<td>Specifies the and criteria in a custom filter preset definition.</td>
</tr>
<tr>
<td>orCriteria</td>
<td>Specifies the or criteria in a custom filter preset definition.</td>
</tr>
<tr>
<td>componentEventsFile</td>
<td>Specifies a component definition file under the 11g 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 11g Release 1 (11.1.1.6) metadata model component, and the filter preset level is set to “Custom”.</td>
</tr>
</tbody>
</table>
2.1.2.4 getAuditRepository
Online command that displays audit repository settings.

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.

Syntax
getAuditRepository

Example
The following example 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.

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

Syntax
setAuditRepository([switchToDB],[dataSourceName],[interval],
[timezone],[repositoryType],[logDirectory],
[jdbcString],[dbUser],[dbPassword])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>switchToDB</td>
<td>If true, switches the repository from file to database. Valid value: true.</td>
</tr>
<tr>
<td>dataSourceName</td>
<td>Specifies the JNDI name of the data source. This data source must be configured in the specified Oracle Weblogic Server domain.</td>
</tr>
<tr>
<td>interval</td>
<td>Specifies the time, in seconds, that the audit loader sleeps.</td>
</tr>
<tr>
<td>timezone</td>
<td>Specifies the time zone in which the audit loader records the timestamps of the audit events. Valid values are utc and local.</td>
</tr>
<tr>
<td>repositoryType</td>
<td>Specifies the database type to which the data has to be uploaded. The supported databases are Oracle, MS SQL Server and IBM DB2.</td>
</tr>
<tr>
<td>logDirectory</td>
<td>Specifies the audit log directory for SE applications to store bus stop files.</td>
</tr>
</tbody>
</table>
The following example changes audit repository to a specific database and sets the audit loader interval to 14 seconds, and the time zone to utc:

```
setAuditRepository(switchToDB="true",
dataSourceName="jdbc/AuditDB",interval="14",timezone="utc",
repositoryType="DB_ORACLE", logDirectory="/foo",
jdbcStrings="jdbc:oracle:thin:@db.example.com:5001:sid",
dbUser="scott_iu", dbPassword="tiger")
```

### 2.1.2.6 listAuditEvents

**Online command that displays a component's audit events.**

**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.

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbeanName</td>
<td>Specifies the name of the component MBean.</td>
</tr>
<tr>
<td>componentType</td>
<td>Specifies the component type to limit the list to all events of the component type.</td>
</tr>
</tbody>
</table>

**Examples**

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

```
listAuditEvents(componentType='JPS');
```

**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 example displays audit events for Oracle HTTP Server:

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

The following example 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
  HomeInstanceId
  ECID
  RID
  ...

2.1.2.7 exportAuditConfig
Online command that exports a component’s audit configuration.

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.

Syntax
exportAuditConfig([mbeanName],fileName, [componentType])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mbeanName</td>
<td>Specifies the name of the non-Java EE component MBean.</td>
</tr>
</tbody>
</table>
Example
The following example exports the audit configuration for a component:

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

The following example 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.

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.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mbeanName</code></td>
<td>Specifies the name of the non-Java EE component MBean.</td>
</tr>
<tr>
<td><code>fileName</code></td>
<td>Specifies the path and file name from which the audit configuration</td>
</tr>
<tr>
<td></td>
<td>should be imported.</td>
</tr>
<tr>
<td><code>componentType</code></td>
<td>Specifies that only events of the given component be imported from the file.</td>
</tr>
<tr>
<td></td>
<td>If not specified, the audit configuration in <code>jps-config.xml</code> is imported.</td>
</tr>
</tbody>
</table>

Examples
The following example 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 example 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 generates a view for audit in the database.

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 directory), and the view is created using the IAU_BASE and component tables.

Syntax

createAuditDBView(fileName, componentType, [dbType], [viewType])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>The path and file name to which the SQL script is written.</td>
</tr>
<tr>
<td>componentType</td>
<td>The name of the registered component.</td>
</tr>
<tr>
<td>dbType</td>
<td>The database type. One of the following: DB_ORACLE, MS_SQL_SERVER, IBM_DB2.</td>
</tr>
<tr>
<td>viewType</td>
<td>The view type. One of the following: SIMPLE, INDEXABLE.</td>
</tr>
</tbody>
</table>

Example

wls:/mydomain/serverConfig>
createAuditDBView(fileName="/tmp/JPSAuditView.sql", componentType="JPS",
            dbType="DB_ORACLE", viewType=INDEXABLE)

2.1.2.10 createIAUView

Generates an SQL script to create an IAU view in the database.

Description

The generated script creates, by default, a SIMPLE view when the component is registered with the audit service; it switches the view from SIMPLE to INDEXABLE, or creates a view in the database. INDEXABLE views are supported for an Oracle database only. SIMPLE views can be created for all supported databases in the IAU_VIEWER schema.

Syntax

createIAUView(componentType, [viewType])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>componentType</td>
<td>The component whose definitions are the basis of the view.</td>
</tr>
</tbody>
</table>
Examples

wls:/mydomain/serverConfig>createIAUView(componentType="AuditApp, viewType="INDEXABLE")

wls:/mydomain/serverConfig>createIAUView(componentType="AuditApp, viewType="SIMPLE")

wls:/mydomain/serverConfig>createIAUView(componentType="AuditApp")

2.1.2.11 getIAUViewInfo

Returns information about the view of a component.

Description

Retrieves information about the view of a specified component.

Syntax

getIAUViewInfo(componentType)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>componentType</td>
<td>The component whose definitions are the basis of the view.</td>
</tr>
</tbody>
</table>

Example

wls:/mydomain/serverConfig> getIAUViewInfo(componentType="JPS")

2.1.2.12 listAuditComponents

Lists components that can be audited.

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.

Syntax

listAuditComponents(fileName)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>Specifies the path and file name to which the output is written.</td>
</tr>
</tbody>
</table>

Example

listAuditComponents(fileName = "/tmp/complist.txt")

2.1.2.13 registerAudit

Registers a component with the audit service.
**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.

**Syntax**

```
registerAudit(xmlFile, [xlfFile], componentType, [mode=OVERWRITE|UPGRADE], [createView=SIMPLE|INDEXABLE|DISABLE])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlFile</td>
<td>Specifies the Component Event definition file.</td>
</tr>
<tr>
<td>xlfFile</td>
<td>Specifies the component xlf jar file. Optional.</td>
</tr>
<tr>
<td>componentType</td>
<td>Specifies the component to be registered.</td>
</tr>
<tr>
<td>mode</td>
<td>Optional. OVERWRITE or UPGRADE. Default is UPGRADE.</td>
</tr>
<tr>
<td>createView</td>
<td>Optional. SIMPLE, INDEXABLE or DISABLE. Default is SIMPLE.</td>
</tr>
</tbody>
</table>

**Example**

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

### 2.1.2.14 deregisterAudit

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

**Description**

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

**Syntax**

```
deregisterAudit(componentType)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>componentType</td>
<td>Specifies the component whose definitions are to be removed.</td>
</tr>
</tbody>
</table>

**Example**

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

<table>
<thead>
<tr>
<th>Use this Command...</th>
<th>to...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>changeKeyPassword</td>
<td>Change the password for a key.</td>
<td>Online</td>
</tr>
<tr>
<td>changeKeyStorePassword</td>
<td>Change the password on a keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>createKeyStore</td>
<td>Create a keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteKeyStore</td>
<td>Delete a keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteKeyStoreEntry</td>
<td>Delete an entry in a keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>exportKeyStore</td>
<td>Export a keystore to file.</td>
<td>Online</td>
</tr>
<tr>
<td>exportKeyStoreCertificate</td>
<td>Export a certificate to a file.</td>
<td>Online</td>
</tr>
<tr>
<td>exportKeyStoreCertificateRequest</td>
<td>Export a certificate request to a file.</td>
<td>Online</td>
</tr>
<tr>
<td>generateKeyPair</td>
<td>Generate a keypair.</td>
<td>Online</td>
</tr>
<tr>
<td>generateSecretKey</td>
<td>Generate a secret key.</td>
<td>Online</td>
</tr>
<tr>
<td>getKeyStoreCertificates</td>
<td>Get information about a certificate or trusted certificate.</td>
<td>Online</td>
</tr>
<tr>
<td>getKeyStoreSecretKeyProperties</td>
<td>Get the secret key properties.</td>
<td>Online</td>
</tr>
<tr>
<td>importKeyStore</td>
<td>Import a keystore from file.</td>
<td>Online</td>
</tr>
<tr>
<td>importKeyStoreCertificate</td>
<td>Import a certificate or other object.</td>
<td>Online</td>
</tr>
<tr>
<td>listExpiringCertificates</td>
<td>List certificates expiring in a specified period.</td>
<td>Online</td>
</tr>
<tr>
<td>listKeyStoreAliases</td>
<td>List aliases in a keystore.</td>
<td>Online</td>
</tr>
<tr>
<td>listKeyStores</td>
<td>List all the keystores in a stripe.</td>
<td>Online</td>
</tr>
<tr>
<td>syncKeyStores</td>
<td>Synchronizes the keystores in the administration server with keystores in the security store.</td>
<td>Online</td>
</tr>
</tbody>
</table>

### 2.1.3.1 changeKeyPassword

Changes a key password.

**Description**

Changes the password for a key.

**Syntax**

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

**Argument**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc</td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td>appStripe</td>
<td>Specifies the name of the stripe containing the keystore</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the keystore</td>
</tr>
</tbody>
</table>
### Example
The following example changes the password on the key entry `orakey`:

```plaintext
wlst:/mydomain/serverConfig> changeKeyPassword(appStripe='system',
name='keystore', password='password',
alias='orakey', currentkeypassword='currentkeypassword',
newkeypassword='newkeypassword')
```

### 2.1.3.2 changeKeyStorePassword
Changes the password of a keystore.

#### Description
Changes the password of the specified keystore.

#### Syntax
```plaintext
changeKeyStorePassword(appStripe='stripe', name='keystore',
currentpassword='currentpassword', newpassword='newpassword')
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appStripe</code></td>
<td>Specifies the name of the stripe containing the keystore</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Specifies the name of the keystore</td>
</tr>
<tr>
<td><code>currentpassword</code></td>
<td>Specifies the current keystore password</td>
</tr>
<tr>
<td><code>newpassword</code></td>
<td>Specifies the new keystore password</td>
</tr>
</tbody>
</table>

#### Example
The following example changes the password for `keystore2`.

```plaintext
wlst:/mydomain/serverConfig> changeKeyStorePassword(appStripe='system',
name='keystore2', currentpassword='currentpassword', newpassword='newpassword')
```

### 2.1.3.3 createKeyStore
This keystore service command creates a new keystore.

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

#### Syntax
```plaintext
createKeyStore(appStripe='stripe', name='keystore',
password='password', permission=true|false)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>appStripe</code></td>
<td>Specifies the name of the stripe containing the keystore</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Specifies the name of the keystore</td>
</tr>
<tr>
<td><code>password</code></td>
<td>Specifies the keystore password</td>
</tr>
<tr>
<td><code>permission</code></td>
<td>Specifies the permission to create the keystore</td>
</tr>
</tbody>
</table>
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.

**Example**
The following example creates a keystore named keystore1.

```
wlst:/mydomain/serverConfig> createKeyStore(appStripe='system', name='keystore1', password='password', permission=true)
```

### 2.1.3.4 deleteKeyStore

**Deleting the named keystore.**

**Description**
This keystore service command deletes a specified keystore.

**Syntax**
deleteKeyStore(appStripe='stripe', name='keystore', password='password')

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
</table>
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. |

**Example**
The following example deletes the keystore named keystore1.

```
wlst:/mydomain/serverConfig> deleteKeyStore(appStripe='system', name='keystore1', password='password')
```

### 2.1.3.5 deleteKeyStoreEntry

**Deleting a keystore entry.**

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

**Syntax**
deleteKeyStoreEntry(appStripe='stripe', name='keystore', password='password', alias='alias', keypassword='keypassword')

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
</table>
svc                 | Specifies the service command object obtained through a call to getOpssService(). |
appStripe           | Specifies the name of the stripe where the keystore resides. |
name                | Specifies the name of the keystore. |
password            | Specifies the keystore password. |
alias               | Specifies the alias of the entry to be deleted. |
keypassword         | Specifies the key password of the entry to be deleted. |
### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc</td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td>appStripe</td>
<td>Specifies the name of the stripe where the keystore resides.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the keystore.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the keystore password.</td>
</tr>
<tr>
<td>alias</td>
<td>Specifies the alias of the entry to be deleted</td>
</tr>
<tr>
<td>keypassword</td>
<td>Specifies the key password of the entry to be deleted</td>
</tr>
</tbody>
</table>

### Example

The following example deletes a keystore entry denoted by alias orakey.

```
wls:/mydomain/serverConfig> deleteKeyStoreEntry(appStripe='system', name='keystore2', password='password', alias='orakey', keypassword='keypassword')
```

### 2.1.3.6 exportKeyStore

Exports a keystore to a file.

#### Description

Exports a keystore to a specified file.

#### Syntax

```
exportKeyStore(appStripe='stripe', name='keystore', password='password', aliases='comma-separated-aliases', keypasswords='comma-separated-keypasswords', type='keystore-type', filepath='absolute_file_path')
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc</td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td>appStripe</td>
<td>Specifies the name of the stripe where the keystore resides.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the keystore.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the keystore password. The value also applies to the output file, based on the current usage of the command:</td>
</tr>
<tr>
<td></td>
<td>- For password-protected keystores of all types, this will be the password of the output file;</td>
</tr>
<tr>
<td></td>
<td>- For permission-protected keystores of type JKS or JCEKS, this will be the password of the output file;</td>
</tr>
<tr>
<td></td>
<td>- 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.</td>
</tr>
<tr>
<td></td>
<td>If the keystore is password-based, the value of this argument must be the same as the password specified when the password-based keystore was created. Otherwise, if the keystore is not password-based, any value is valid.</td>
</tr>
<tr>
<td>aliases</td>
<td>Specifies a comma separated list of aliases to be exported.</td>
</tr>
</tbody>
</table>
Infrastructure Commands

Infrastructure Security Custom WLST Commands

Examples

The following example exports two aliases from the specified keystore.

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

The following example exports a keystore to create an Oracle Wallet file:

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

2.1.3.7 exportKeyStoreCertificate

Exports a certificate.

Description

Exports a certificate, trusted certificate or certificate chain.

Syntax

```
exportKeyStoreCertificate(appStripe='stripe', name='keystore',
password='password',alias='alias', keypassword='keypassword',
type='entrytype',filepath='absolute_file_path')
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc</td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td>appStripe</td>
<td>Specifies the name of the stripe where the keystore resides.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the keystore.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the keystore password.</td>
</tr>
<tr>
<td>alias</td>
<td>Specifies the alias of the entry to be exported</td>
</tr>
<tr>
<td>keypassword</td>
<td>Specifies the key password.</td>
</tr>
</tbody>
</table>
Infrastructure Commands

2.1.3.8 exportKeyStoreCertificateRequest

Exports a certificate request.

Description
Generates and exports a certificate request from a keystore.

Syntax
exportKeyStoreCertificateRequest(appStripe='stripe', name='keystore', password='password', alias='alias', keypassword='keypassword', filepath='absolute_file_path')

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc</td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td>appStripe</td>
<td>Specifies the name of the stripe where the keystore resides.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the keystore.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the keystore password.</td>
</tr>
<tr>
<td>alias</td>
<td>Specifies the entry's alias name.</td>
</tr>
<tr>
<td>keypassword</td>
<td>Specifies the key password.</td>
</tr>
<tr>
<td>filepath</td>
<td>Specifies the absolute path of the file where certificate request is exported.</td>
</tr>
</tbody>
</table>

Example
The following example exports a certificate request corresponding to the orakey alias:

```
ws:/mydomain/serverConfig> exportKeyStoreCertificateRequest(appStripe='system', name='keystore2', password='password', alias='orakey', keypassword='keypassword', type='Certificate', filepath='/tmp/cert.txt')
```

2.1.3.9 generateKeyPair

Generates a key pair in a keystore.

Description
Generates a key pair using a specified algorithm, and wraps it in a demo CA-signed certificate.
Syntax

generateKeyPair(appStripe='stripe', name='keystore', password='password',
dn='distinshedname', keysize='keysize', alias='alias',
keypassword='keypassword'[, algorithm='algoName'])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc</td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td>appStripe</td>
<td>Specifies the name of the stripe where the keystore resides.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the keystore.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the keystore password.</td>
</tr>
<tr>
<td>dn</td>
<td>Specifies the distinguished name of the certificate wrapping the key pair.</td>
</tr>
<tr>
<td>keysize</td>
<td>Specifies the key size.</td>
</tr>
<tr>
<td>alias</td>
<td>Specifies the alias of the key pair entry.</td>
</tr>
<tr>
<td>keypassword</td>
<td>Specifies the key password.</td>
</tr>
<tr>
<td>algorithm</td>
<td>Specifies the algorithm to use to encrypt the generated keys. The only valid values are RSA or EC (Elliptic Curve Cryptography). Optional. If not specified, the command used the RSA algorithm.</td>
</tr>
</tbody>
</table>

Examples

The following example generates a keypair in keystore2 using the default RSA algorithm:

`wls:/mydomain/serverConfig> generateKeyPair(appStripe='system', name='keystore2', password='password', dn='cn=www.oracle.com', keysize='1024', alias='orakey', keypassword='keypassword')`

The following example generates a keypair in keystore2 using the RSA algorithm:

`wls:/mydomain/serverConfig> generateKeyPair(appStripe='system', name='keystore2', password='password', dn='cn=www.oracle.com', keysize='1024', alias='orakey', keypassword='keypassword', algorithm='RSA')`

The following example generates a keypair in keystore2 using the ECC (Elliptic Curve Cryptography) algorithm:

`wls:/mydomain/serverConfig> generateKeyPair(appStripe='system', name='keystore2', password='password', dn='cn=www.oracle.com', keysize='1024', alias='orakey', keypassword='keypassword', algorithm='EC')`

2.1.3.10 generateSecretKey

Generates a secret key.

Description

Generates a symmetric key in a keystore.

Syntax

generateSecretKey(appStripe='stripe', name='keystore', password='password',
algorithm='algorithm', keysize='keysize', alias='alias',
keypassword='keypassword')
### Argument Definition

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
</table>
| 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. |

### Example

The following example generates a keypair with keysize 128 in `keystore2`.

```
wlst:/mydomain/serverConfig> 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.**

**Description**

Retrieves information about a certificate or trusted certificate.

**Syntax**

```
getKeyStoreCertificates(appStripe='stripe', name='keystore', password='password', alias='alias', keypassword='keypassword')
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
</table>
| 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. |

**Example**

The following example gets certificates associated with `keystore3`.

```
wlst:/mydomain/serverConfig> getKeyStoreCertificates(appStripe='system', name='keystore3', password='password', alias='orakey', keypassword='keypassword')
```

#### 2.1.3.12 getKeyStoreSecretKeyProperties

**Retrieves secret key properties.**
Description
Retrieves secret key properties like the algorithm.

Syntax
getKeyStoreSecretKeyProperties(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword')

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc</td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td>appStripe</td>
<td>Specifies the name of the stripe where the keystore resides.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the keystore.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the keystore password.</td>
</tr>
<tr>
<td>alias</td>
<td>Specifies the alias of the secret key whose properties are displayed.</td>
</tr>
<tr>
<td>keypassword</td>
<td>Specifies the secret key password.</td>
</tr>
</tbody>
</table>

Example
The following example gets properties for secret key seckey:

```
wlst:/mydomain/serverConfig> getKeyStoreSecretKeyProperties(appStripe='system',
name='keystore3',
password='password', alias='seckey', keypassword='keypassword')
```

2.1.3.13 importKeyStore
Imports a keystore from file.

Description
Imports a keystore from a system file.

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')

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc</td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td>appStripe</td>
<td>Specifies the name of the stripe where the keystore will reside.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the keystore.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the keystore password. These rules apply:</td>
</tr>
<tr>
<td></td>
<td>■ If importing an auto-login Oracle Wallet file, no password is needed.</td>
</tr>
<tr>
<td></td>
<td>■ If importing a password-protected Oracle Wallet file (ewallet.p12), enter a password of minimum eight characters.</td>
</tr>
<tr>
<td>aliases</td>
<td>Specifies the comma-separated aliases of the entries to be imported from the file. If type is set to OracleWallet, it is not required; otherwise, it is a required argument.</td>
</tr>
</tbody>
</table>
Example

The following example imports a JKS keystore file to keystore2:

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

The following 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.

Description

Imports a certificate, trusted certificate or certificate chain.

Syntax

```
importKeyStoreCertificate(appStripe='stripe', name='keystore', password='password', alias='alias', keypassword='keypassword', type='entrytype',filepath='absolute_file_path')
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>svc</td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td>appStripe</td>
<td>Specifies the name of the stripe where the keystore resides.</td>
</tr>
<tr>
<td>name</td>
<td>Specifies the name of the keystore.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the keystore password.</td>
</tr>
<tr>
<td>alias</td>
<td>Specifies the alias of the entry to be imported.</td>
</tr>
<tr>
<td>keypassword</td>
<td>Specifies the key password of the newly imported entry.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of keystore entry to be imported. Valid values are 'Certificate', 'TrustedCertificate' or 'CertificateChain'.</td>
</tr>
</tbody>
</table>
Example

The following example imports a certificate into keystore2.

```java
wls:/mydomain/serverConfig> 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.

Description

Lists expiring certificates and optionally renews them.

Syntax

```java
listExpiringCertificates(days='days', autorenew=true|false)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>svc</code></td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td><code>days</code></td>
<td>Specifies that the list should only include certificates within this many days from expiration.</td>
</tr>
<tr>
<td><code>autorenew</code></td>
<td>Specifies true for automatically renewing expiring certificates, false for only listing them.</td>
</tr>
</tbody>
</table>

Example

The following example lists certificates expiring within one year, and requests that they be renewed:

```java
wls:/mydomain/serverConfig> listExpiringCertificates(days='365', autorenew=true)
```

2.1.3.16 `listKeyStoreAliases`

Lists the aliases in a keystore.

Description

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

Syntax

```java
listKeyStoreAliases(appStripe='stripe', name='keystore', password='password', type='entrytype')
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>svc</code></td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td><code>appStripe</code></td>
<td>Specifies the name of the stripe where the keystore resides.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Specifies the name of the keystore.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>svc</code></td>
<td>Specifies the service command object obtained through a call to getOpssService().</td>
</tr>
<tr>
<td><code>appStripe</code></td>
<td>Specifies the name of the stripe where the keystore resides.</td>
</tr>
<tr>
<td><code>name</code></td>
<td>Specifies the name of the keystore.</td>
</tr>
</tbody>
</table>
### Example

The following example lists secret keys in keystore2:

```bash
wls:/mydomain/serverConfig> listKeyStoreAliases(appStripe='system',
name='keystore2',
password='password', type='SecretKey')
```

#### 2.1.3.17 listKeyStores

Lists all the keystores in a stripe.

**Description**

Lists all the keystores in the specified stripe.

**Syntax**

```bash
listKeyStores(appStripe='stripe')
```

**Argument Definition**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>svc</code></td>
<td>Specifies the service command object obtained through a call to <code>getOpssService()</code></td>
</tr>
<tr>
<td><code>appStripe</code></td>
<td>Specifies the name of the stripe whose keystores are listed.</td>
</tr>
</tbody>
</table>

**Example**

The following example lists all keystores on all stripes.

```bash
wls:/mydomain/serverConfig> 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.

**Description**

Synchronizes keystores in the central security store with those present in the domain `config` 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.

**Syntax**

The syntax is as follows:

```bash
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.

---

### Example

The following example 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:

```java
wls:/mydomain/serverConfig> syncKeyStores()
```

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

```java
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**

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>activateIDSConfigChanges</td>
<td>Reload the Identity Directory Service configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addAttributeInEntityConfig</td>
<td>Add a new attribute to the entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addAttributePropsInEntityConfig</td>
<td>Add new properties for an attribute in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addAttributeRefForEntity</td>
<td>Add a new attribute to the specified entity.</td>
<td>Online</td>
</tr>
<tr>
<td>addAttrrefPropsInEntityConfig</td>
<td>Add new properties for an attribute reference in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addCommonPropertyForOperationConfig</td>
<td>Add a new property for a specified operation configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addEntity</td>
<td>Add a new entity to the entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addEntityProps</td>
<td>Add new properties for an entity in an entity configuration.</td>
<td>Online</td>
</tr>
</tbody>
</table>
Table 2–5 (Cont.) WLST Identity Directory Service Commands

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addEntityRelation</td>
<td>Add a new entity relation to the entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addIdentityDirectoryService</td>
<td>Add a new Identity Directory Service to the configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addOperationConfig</td>
<td>Add a new operation configuration to the entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addPropertyForOperationConfig</td>
<td>Add a new property to a specified operation configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteAttributeInEntityConfig</td>
<td>Delete an attribute from an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteAttributePropsInEntityConfig</td>
<td>Delete attribute properties in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteAttrrefPropsInEntityConfig</td>
<td>Delete attribute reference properties in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteEntity</td>
<td>Delete an entity from an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteEntityProps</td>
<td>Delete entity properties in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteEntityRelation</td>
<td>Delete the specified entity relation.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteIdentityDirectoryService</td>
<td>Delete the specified Identity Directory Service in the configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteOperationConfig</td>
<td>Delete operation configuration in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>listAllAttributeInEntityConfig</td>
<td>List all attributes in the entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>listAllEntityInEntityConfig</td>
<td>List all entities defined in the specified entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>listAllIdentityDirectoryService</td>
<td>List all Identity Directory Services in the configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>removeAttributeRefForEntity</td>
<td>Remove an attribute from the specified entity.</td>
<td>Online</td>
</tr>
<tr>
<td>removeCommonPropertyForOperationConfig</td>
<td>Remove a property for the specified operation configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>removePropertyForOperationConfig</td>
<td>Remove a property for the specified operation configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>updateAttributeInEntityConfig</td>
<td>Update attributes in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>updateAttributePropsInEntityConfig</td>
<td>Update attribute properties in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>updateAttrrefPropsInEntityConfig</td>
<td>Update attribute reference properties in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>updateEntity</td>
<td>Update an entity’s properties in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>updateEntityAttrs</td>
<td>Update an entity’s properties in an entity configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>updateEntityProps</td>
<td>Update the entity properties in an entity configuration.</td>
<td>Online</td>
</tr>
</tbody>
</table>
2.1.4.1 activateIDSConfigChanges
Online command that reloads the configuration for Identity Directory Service.

Description
Reloads the Identity Directory Service configuration.

Syntax
activateIDSConfigChanges()

This command has no arguments.

Example
The following command reloads the Identity Directory Service configuration:
activateIDSConfigChanges()

2.1.4.2 addAttributeInEntityConfig
Online command that adds an attribute to the entity configuration.

Description
Adds a new attribute to the entity configuration.

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

Table 2–6  addAttributeInEntityConfig Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the attribute to be added.</td>
</tr>
</tbody>
</table>
Example
The following command adds an attribute commonname of userrole entity:

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

2.1.4.3 addAttributePropsInEntityConfig
Online command that adds properties for an attribute in an entity configuration.

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the attribute to be added.</td>
</tr>
<tr>
<td>propNames</td>
<td>List of property names separated by &quot;</td>
</tr>
<tr>
<td>propVals</td>
<td>List of corresponding property values separated by &quot;</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

Example
The following command adds an attribute commonname of userrole entity:

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

2.1.4.3 addAttributePropsInEntityConfig
Online command that adds properties for an attribute in an entity configuration.

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the attribute to be added.</td>
</tr>
<tr>
<td>propNames</td>
<td>List of property names separated by &quot;</td>
</tr>
<tr>
<td>propVals</td>
<td>List of corresponding property values separated by &quot;</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>
Example
The following command adds an attribute **orgunit** of entity **userrole**:

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

### 2.1.4.4 addAttributeRefForEntity
Online command that adds attribute to an entity.

**Description**
Adds a new attribute to the specified entity.

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

**Table 2–8 addAttributeRefForEntity Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the entity to which the attribute will be added.</td>
</tr>
<tr>
<td>attrRefName</td>
<td>Name of the attribute to be added to the entity.</td>
</tr>
</tbody>
</table>
| 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.                                  |

Example
The following command adds an attribute **User** to **userrole** entity:

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

### 2.1.4.5 addAttrrefPropsInEntityConfig
Online command that adds property for an attribute reference.
**Description**

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

**Syntax**

```java
addAttrrefPropsInEntityConfig(entityName, attrName, propNames, propVals, appName)
```

**Table 2–9  addAttrrefPropsInEntityConfig Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>Name of the entity.</td>
</tr>
<tr>
<td>attrName</td>
<td>Name of the attribute reference.</td>
</tr>
</tbody>
</table>
| 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.                                     |

**Example**

The following command adds a multivalued property `labelname` for `org` entity:

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

**2.1.4.6 addCommonPropertyForOperationConfig**

Online command that adds a property for an operation configuration.

**Description**

Adds a new property for a specified operation configuration.

**Syntax**

```java
addCommonPropertyForOperationConfig(entityName, propName, propValue, appName)
```

**Table 2–10  addCommonPropertyForOperationConfig Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>Name of the entity.</td>
</tr>
<tr>
<td>propName</td>
<td>Name of the property to be added for this operation configuration.</td>
</tr>
<tr>
<td>propValue</td>
<td>Value of the property to be added for this operation configuration.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>
Example
The following command adds a new property member:
addCommonPropertyForOperationConfig('groupmember.attr', 'member', 'userrole')

2.1.4.7 addEntity
Online command that adds an entity to the configuration.

Description
Adds a new entity to the entity configuration.

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

Table 2–11  addEntity Arguments

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

Example
The following command adds an attribute group to the Group entity.
addEntity('Group','group','commonname',true,true,true,true,'name|commonname','none
|none|true|false|userrole|

### 2.1.4.8 addEntityProps

**Add**s property for an entity.

**Description**

Online command that adds new properties for an entity in an entity configuration.

**Syntax**

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

**Table 2–12  addEntityProps Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the entity.</td>
</tr>
<tr>
<td>propNames</td>
<td>List of property names separated by &quot;</td>
</tr>
<tr>
<td>propValues</td>
<td>List of corresponding property values separated by &quot;</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

**Example**

The following command adds `inclobjclasses` and `exclobjclasses` properties:

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

### 2.1.4.9 addEntityRelation

Online command that **adds** entity relation to an entity.

**Description**

Add a new entity relation to the entity configuration for the specified attributes.

**Syntax**

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

**Table 2–13  addEntityRelation Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the relation between the entities for the given attributes.</td>
</tr>
<tr>
<td>type</td>
<td>Type of the entity relation (&quot;ManyToMany&quot;, &quot;ManyToOne&quot;, &quot;OneToMany&quot;, &quot;OneToOne&quot;).</td>
</tr>
<tr>
<td>fromEntity</td>
<td>Name of the from entity.</td>
</tr>
<tr>
<td>fromAttr</td>
<td>Name of the from attribute.</td>
</tr>
<tr>
<td>toEntity</td>
<td>Name of the to entity.</td>
</tr>
<tr>
<td>toAttr</td>
<td>Name of the to attribute.</td>
</tr>
<tr>
<td>recursive</td>
<td>Flag to set the entity relationship as recursive.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>
Example
The following command adds the manager relation between the manager and User entities:

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

### 2.1.4.10 addIdentityDirectoryService
Online command that adds an Identity Store Service.

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

**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. |

Example
The following command adds the userrole IdentityStoreService:

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

### 2.1.4.11 addOperationConfig
Online command that adds operation configuration to an entity.

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

**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. |
Example
The following command adds the User entity to which the operation configuration will be added:

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

### 2.1.4.12 addPropertyForOperationConfig
Online command that adds a property to an operation configuration.

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

#### Syntax
```java
addPropertyForOperationConfig(entityName, propName, propValue, appName)
```

#### Table 2–16 addPropertyForOperationConfig Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>Name of the entity to which the operation configuration will be added.</td>
</tr>
<tr>
<td>propName</td>
<td>A property name to be added to the operation configuration.</td>
</tr>
<tr>
<td>propValue</td>
<td>A value for the property added to the operation configuration.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

Example
The following command adds the property to the operation configuration:

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

### 2.1.4.13 deleteAttributeInEntityConfig
Online command that deletes attribute from an entity.

#### Description
Deletes an attribute from an entity configuration.

#### Syntax
```java
deleteAttributeInEntityConfig(name, appName)
```

#### Table 2–17 deleteAttributeInEntityConfig Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the attribute to be deleted.</td>
</tr>
</tbody>
</table>
Example
The following command deletes the `commonname` attribute.
```
deleteAttributeInEntityConfig('commonname', 'userrole')
```

### 2.1.4.14 `deleteAttributePropsInEntityConfig`

Online command that deletes the properties of an attribute.

**Description**
Deletes attribute properties in an entity configuration.

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

#### Table 2–18  `deleteAttributePropsInEntityConfig` Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>name</code></td>
<td>Name of the attribute.</td>
</tr>
<tr>
<td><code>propNames</code></td>
<td>List of property names separated by &quot;</td>
</tr>
<tr>
<td><code>appName</code></td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

Example
The following example deletes the property `labelname` from the `userrole` entity:
```
deleteAttributePropsInEntityConfig('orgunit','labelname|multivalued','userrole')
```

### 2.1.4.15 `deleteAttrrefPropsInEntityConfig`

Online command that deletes attribute reference properties in an entity.

**Description**
Deletes one or more attribute reference properties in an entity configuration.

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

#### Table 2–19  `deleteAttrrefPropsInEntityConfig` Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>entityName</code></td>
<td>Name of the entity.</td>
</tr>
<tr>
<td><code>attrName</code></td>
<td>Name of the attribute reference.</td>
</tr>
<tr>
<td><code>propNames</code></td>
<td>List of property names to be deleted. If multiple properties are to be deleted, they should be separated by &quot;</td>
</tr>
<tr>
<td><code>appName</code></td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>
**Example**
The following command deletes two properties from attribute reference `orgunit` of entity `org`:

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

### 2.1.4.16 deleteEntity
Online command that deletes an entity.

**Description**
Deletes an entity from an entity configuration.

**Syntax**
```
deleteEntity(name, appName)
```

**Table 2–20  deleteEntity Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the entity to be deleted.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

**Example**
The following command deletes the `User` entity.
```
deleteEntity('User', 'userrole')
```

### 2.1.4.17 deleteEntityProps
Online command that deletes the properties of an entity.

**Description**
Deletes entity properties in an entity configuration.

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

**Table 2–21  deleteEntityProps Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the entity.</td>
</tr>
<tr>
<td>propNames</td>
<td>List of property names separated by &quot;</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

**Example**
The following command deletes the two properties `incobjclasses` and `excobjclasses` of `User` entity:
```
deleteEntityProps('User','incobjclasses|excobjclasses','userrole')
```
2.1.4.18 deleteEntityRelation
Online command that deletes the relationship between entities.

Description
Deletes the specified entity relation between entities for the given attributes.

Syntax
deleteEntityRelation(name, appName)

Table 2–22 deleteEntityRelation Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the relation between the entities for the given attributes.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

Example
The following command deletes the manager relation specified between entities:
deleteEntityRelation('manager', 'userrole')

2.1.4.19 deleteIdentityDirectoryService
Online command that deletes the specified IdentityStoreService.

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

Syntax
deleteIdentityDirectoryService(name)

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

Example
The following example deletes ids1 IdentityStoreService configuration.
deleteIdentityDirectoryService('ids1')

2.1.4.20 deleteOperationConfig
Online command that deletes an operation configuration.

Description
Deletes an operation configuration in an entity configuration.

Syntax
deleteOperationConfig(entityName, appName)
Table 2–23  deleteOperationConfig Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>Name of the entity from which the operation configuration will be removed.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

Example

The following command deletes the operation configuration associated with entity User and application userrole:

deleatOperationConfig('User','userrole')

2.1.4.21 listAllAttributeInEntityConfig

Online command that lists all attributes.

Description

Lists all attributes in the entity configuration.

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.

Example

The following command obtains the list of attributes from userrole entity:

listAllAttributeInEntityConfig('userrole')

2.1.4.22 listAllEntityInEntityConfig

Online command that lists all entities for an entity configuration.

Description

Lists all entities defined in the specified entity configuration.

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.

Example

The following command obtains the list of entities associated with userrole entity:

listAllEntityInEntityConfig('userrole')

2.1.4.23 listAllIdentityDirectoryService

Online command that lists all IdentityStoreService for an Identity Directory Service configuration.
**Description**  
Lists all IdentityStoreService in Identity Directory Service configuration.

**Syntax**  
listAllIdentityDirectoryService()

This command has no arguments.

**Example**  
The following command lists all the IdentityStoreService for an Identity Directory Service configuration:

listAllIdentityDirectoryService()

### 2.1.4.24 `removeAttributeRefForEntity`  
Online command that deletes an attribute from an entity.

**Description**  
Removes an attribute from the specified entity.

**Syntax**  
removeAttributeRefForEntity(name, attrRefName, appName)

<table>
<thead>
<tr>
<th>Table 2–24 <code>removeAttributeRefForEntity</code> Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Argument</strong></td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>attrRefName</td>
</tr>
<tr>
<td>appName</td>
</tr>
</tbody>
</table>

**Example**  
The following command deletes the `givenname` attribute associated with `User` entity:

removeAttributeRefForEntity('User','givenname','userrole')

### 2.1.4.25 `removeCommonPropertyForOperationConfig`  
Online command that deletes a property for an operation configuration.

**Description**  
Removes a property for the specified operation configuration.

**Syntax**  
removeCommonPropertyForOperationConfig(entityName, propName, appName)

<table>
<thead>
<tr>
<th>Table 2–25 <code>removeCommonPropertyForOperationConfig</code> Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Argument</strong></td>
</tr>
<tr>
<td>entityName</td>
</tr>
</tbody>
</table>
The following command removes `groupmember.attr` property associated with `User` entity:

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

2.1.4.26 removePropertyForOperationConfig

Online command that removes a property for an operation configuration.

**Description**

Removes a property for the specified operation configuration.

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>propName</td>
<td>Name of property to be removed for this operation configuration.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

**Example**

The following command removes `entity.searchbase` property associated with `User` entity:

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

2.1.4.27 updateAttributeInEntityConfig

Online command that updates an attribute for an entity configuration.

**Description**

Updates attributes in an entity configuration.

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the entity attribute to be updated.</td>
</tr>
</tbody>
</table>
Example

The following command updates the **commonname** attribute:

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

### 2.1.4.28 updateAttributePropsInEntityConfig

Online command that updates the properties of an attribute for an entity.

**Description**

Updates attribute properties in an entity configuration.

**Syntax**

```plaintext
updateAttributePropsInEntityConfig(name, propNames, propVals, appName)
```

**Table 2–28  updateAttributePropsInEntityConfig Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the attribute to be updated.</td>
</tr>
<tr>
<td>propNames</td>
<td>List of property names separated by &quot;</td>
</tr>
<tr>
<td>propVals</td>
<td>List of corresponding property values separated by &quot;</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

**Example**

The following command updates the properties for **orgunit** attribute associated with **userrole** application:

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

### 2.1.4.29 updateAttrrefPropsInEntityConfig

Online command that updates attribute reference properties for an entity.

**Description**

Updates attribute reference properties in an entity configuration.
Syntax
updateAttrrefPropsInEntityConfig(entityName, attrName, propNames, propVals, appName)

Table 2–29  updateAttrrefPropsInEntityConfig Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityName</td>
<td>Name of the entity.</td>
</tr>
<tr>
<td>attrName</td>
<td>Name of the attribute reference.</td>
</tr>
<tr>
<td>propNames</td>
<td>List of property names separated by &quot;</td>
</tr>
<tr>
<td>propVals</td>
<td>List of corresponding property values separated by &quot;</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

Example
The following command updates the attribute reference properties:

updateAttrrefPropsInEntityConfig('org', 'orgunit', 'entity.searchbase', 'multivalued', 'userrole')

2.1.4.30 updateEntity
Online command that updates properties of an entity.

Description
Updates an entity’s properties in an entity configuration.

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

Table 2–30  updateEntity Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the entity to be updated.</td>
</tr>
<tr>
<td>type</td>
<td>Type of the entity.</td>
</tr>
<tr>
<td>idAttr</td>
<td>Identity attribute of the entity.</td>
</tr>
<tr>
<td>create</td>
<td>Flag to specify the create is allowed.</td>
</tr>
<tr>
<td>modify</td>
<td>Flag to specify the modify is allowed.</td>
</tr>
<tr>
<td>delete</td>
<td>Flag to specify the delete is allowed.</td>
</tr>
<tr>
<td>search</td>
<td>Flag to specify the search is allowed.</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

Example
The following command updates the properties associated with Group entity:

updateEntity('Group', 'group', 'commonname', true, true, true, true, 'userrole')
2.1.4.31 updateEntityAttrs
Online command that updates the configuration attributes for an entity.

**Description**
Updates the configuration attributes for an entity attribute.

**Syntax**
updateEntityAttrs(name, attrNames, attrVals, appName)

**Table 2–31  updateEntityAttrs Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the entity attribute. To update the properties of an entity attribute, see updateAttributePropsInEntityConfig.</td>
</tr>
<tr>
<td>attrNames</td>
<td>List of configuration attribute names. If multiple configuration attributes are to be updated, they should be separated by &quot;</td>
</tr>
<tr>
<td>attrVals</td>
<td>List of corresponding configuration attribute values separated by &quot;</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

**Example**
The following command updates configuration attributes associated with User entity:
updateEntityAttrs('User','idAttr|firstnameAttr','uid|givenname','userrole')

2.1.4.32 updateEntityProps
Online command that updates the properties of an entity.

**Description**
 Updates the entity properties in an entity configuration.

**Syntax**
updateEntityProps(name, propNames, propVals, appName)
### Table 2–32  `updateEntityProps` Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the attribute to be updated.</td>
</tr>
<tr>
<td>propNames</td>
<td>List of property names separated by &quot;</td>
</tr>
<tr>
<td>propVals</td>
<td>List of corresponding property values separated by &quot;</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

#### Example
The following command updates the properties associated with `User` entity:

```bash
updateEntityProps('User','inclobjclasses|exclobjclasses','inetorgperson|orclidxperson','userrole')
```

#### 2.1.4.33 `deleteAttributePropsInEntityConfig`  
Online command that deletes the attribute properties in an entity configuration.

**Description**  
Deletes the attribute properties in an entity configuration.

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

### Table 2–33  `deleteAttributePropsInEntityConfig`  

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the attribute to be deleted.</td>
</tr>
<tr>
<td>propNames</td>
<td>List of property names separated by &quot;</td>
</tr>
<tr>
<td>appName</td>
<td>Name of the Identity Directory Service.</td>
</tr>
</tbody>
</table>

#### Example
The following command deletes the attribute property, `orgunit` from the `userrole`.  
`deleteAttributePropsInEntityConfig('orgunit','labelname|validvalues','userrole')`

#### 2.1.4.34 `dumpConnectionPoolStatsForInMemoryConfig`  
Online command that dumps the LDAP connection pool statistics for the associated in-memory IDS configuration for the current JVM on which WLS is configured into a specified file.

**Description**  
Dumps the LDAP connection pool statistics for the associated in-memory IDS configuration for the current JVM on which WLS is configured into a specified file.

**Syntax**  
`dumpConnectionPoolStatsForInMemoryConfig(name, fileName)`
Table 2–34  \textit{dumpConnectionPoolStatsForInMemoryConfig}

\begin{tabular}{|l|l|}
\hline
\textbf{Argument} & \textbf{Definition} \\
\hline
name & Name of the in-memory IDS configuration. \\
fileName & Refers to the full path of the file. \\
\hline
\end{tabular}

\textbf{Example}

The following example dumps the connection pool statistics for the in-memory IDS configuration \texttt{idsl} into the specified file:

\begin{verbatim}
dumpConnectionPoolStatsForInMemoryConfig('idsl', '/tmp/dump.txt')
\end{verbatim}

2.1.4.35 \texttt{dumpConnectionPoolStatsForAllInMemoryConfig}

Online command that dumps the LDAP connection pool statistics for all in-memory IDS configuration for the current JVM on which WLS is configured into a specified file.

\textbf{Description}

Dumps the LDAP connection pool statistics for all in-memory IDS configuration for the current JVM on which WLS is configured into a specified file.

\textbf{Syntax}

\begin{verbatim}
dumpConnectionPoolStatsForAllInMemoryConfig(fileName)
\end{verbatim}

Table 2–35  \textit{dumpConnectionPoolStatsForAllInMemoryConfig}

\begin{tabular}{|l|l|}
\hline
\textbf{Argument} & \textbf{Definition} \\
\hline
fileName & Refers to the full path of the file. \\
\hline
\end{tabular}

\textbf{Example}

The following example dumps LDAP connection pool statistics for all in-memory IDS configuration into the specified file:

\begin{verbatim}
dumpConnectionPoolStatsForAllInMemoryConfig('/tmp/dump.txt')
\end{verbatim}

2.1.4.36 \texttt{dumpConnectionPoolStatsForAllFileBasedConfig}

Online command that dumps the LDAP connection pool statistics for all file-based IDS configuration for the current JVM on which WLS is configured into a specified file.

\textbf{Description}

Dumps the LDAP connection pool statistics for all file-based IDS configuration for the current JVM on which WLS is configured into a specified file.

\textbf{Syntax}

\begin{verbatim}
dumpConnectionPoolStatsForAllFileBasedConfig(name, fileName)
\end{verbatim}

Table 2–36  \textit{dumpConnectionPoolStatsForAllFileBasedConfig}

\begin{tabular}{|l|l|}
\hline
\textbf{Argument} & \textbf{Definition} \\
\hline
name & Name of the file-based IDS configuration. \\
\hline
\end{tabular}
2.1.4.37 dumpConnectionPoolStatsForAllFileBasedConfig

Online command that dumps the LDAP connection pool statistics for all file-based IDS configuration in the current JVM on which WLS is configured into a specified file.

**Description**

Dumps the LDAP connection pool statistics for all file-based IDS configuration for the current JVM on which WLS is configured into a specified file.

**Syntax**

dumpConnectionPoolStatsForAllFileBasedConfig(fileName)

**Example**

The following example dumps the connection pool statistics all file-based IDS configuration into the specified file:

dumpConnectionPoolStatsForFileBasedConfig('/tmp/dump.txt')

### Table 2–37 dumpConnectionPoolStatsForAllFileBasedConfig

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>Refers to the full path of the file.</td>
</tr>
</tbody>
</table>

2.1.5 Library Oracle Virtual Directory (libOVD) Commands

Use the WLST commands listed in Table 2–38 to manage a libOVD configuration associated with a specific Oracle Platform Security Services (OPSS) context.

### Table 2–38 WLST libOVD Commands

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addDNAttribute</td>
<td>Add an attribute to the DN attributes list for an existing adapter.</td>
<td>Online</td>
</tr>
<tr>
<td>activateLibOVDConfigChanges</td>
<td>Reload the libOVD configuration.</td>
<td>Online</td>
</tr>
<tr>
<td>addAttributeExclusionRule</td>
<td>Add a attribute exclusion rule.</td>
<td>Online</td>
</tr>
<tr>
<td>addAttributeRule</td>
<td>Add a new attribute mapping rule.</td>
<td>Online</td>
</tr>
<tr>
<td>addDomainExclusionRule</td>
<td>Add a domain exclusion rule.</td>
<td>Online</td>
</tr>
<tr>
<td>addDomainRule</td>
<td>Add a new domain mapping rule.</td>
<td>Online</td>
</tr>
<tr>
<td>addJoinRule</td>
<td>Add a join rule to an existing Join Adapter for a libOVD configuration.</td>
<td>Online</td>
</tr>
</tbody>
</table>
**Table 2–38 (Cont.) WLST libOVD Commands**

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

Online command that adds an attribute to the DN Attributes List.

**Description**

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

**Syntax**

```language
addDNAttribute(adapterName, attributeName, [contextName])
```

**Table 2–39  addDNAttribute Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the adapter to be updated.</td>
</tr>
<tr>
<td>attributeName</td>
<td>Name of the new DN attribute to be added.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is default.</td>
</tr>
</tbody>
</table>

**Example**

The following example adds `memberof` attribute to `ldap1` adapter.
2.1.5.2 activateLibOVDConfigChanges
Online command that reloads the libOVD configuration.

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

**Syntax**
activateLibOVDConfigChanges([contextName])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is default.</td>
</tr>
</tbody>
</table>

**Example**
The following command reloads the default libOVD configuration for a specified OPSS context:
activateLibOVDConfigChanges('default')

2.1.5.3 addAttributeExclusionRule
Online command that adds an attribute exclusion rule.

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

**Syntax**
addAttributeExclusionRule(attribute, mappingContextId, [contextName])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute</td>
<td>Name of the attribute to be added to the exclusion list.</td>
</tr>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is default.</td>
</tr>
</tbody>
</table>

**Example**
The following command adds objectsid to the exclusion list:
addAttributeExclusionRule('objectsid','userrole')

2.1.5.4 addAttributeRule
Online command that adds a new attribute mapping rule.
Description
Adds a new attribute mapping rule to the libOVD configuration associated with a specific OPSS context.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is default.</td>
</tr>
</tbody>
</table>

Example
The following command creates a mapping rule for the libOVD configuration. Here, the lastname is mapped to the cn.
```
addAttributeRule('lastname','','','sn','','','','Inbound','userrole')
```

2.1.5.5 addDomainExclusionRule
Online command that adds a domain exclusion rule.

Description
Adds a domain exclusion rule to the exclusion list.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>Distinguished name (DN) of the attribute to be added to the exclusion list.</td>
</tr>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command adds \(\text{cn=group,dc=oracle,dc=com}\) to the exclusion list:
```
addDomainExclusionRule('cn=group,dc=oracle,dc=com','userrole')
```

2.1.5.6 addDomainRule
Online command that adds a new domain mapping rule.

Description
Adds a new domain mapping rule.
Infrastructure Commands

**Syntax**

```java
addDomainRule(srcDomain, destDomain, domainConstructRule, mappingContextId, [contextName])
```

**Table 2–44 addDomainRule Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>srcDomain</td>
<td>Source domain.</td>
</tr>
<tr>
<td>destDomain</td>
<td>Destination domain</td>
</tr>
<tr>
<td>domainConstructRule</td>
<td>Name of the attribute to be added to the exclusion list.</td>
</tr>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is default.</td>
</tr>
</tbody>
</table>

**Example**

The following command creates a domain mapping rule:

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

### 2.1.5.7 addJoinRule

**Description**

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

**Syntax**

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

**Table 2–45 addJoinRule Arguments**

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

**Examples**

The following commands create different join rules for an existing Join adapter:

```java
addJoinRule('join1', 'secondaryldap', 'cn=cn', 'Simple', 'default')
addJoinRule('join1', 'secondaryldap', 'cn=cn', 'Conditional', 'default')
addJoinRule(adapterName='join1', secondary='LDAP3', condition='uid=cn',`
Infrastructure Commands

JoinerType='OneToMany'

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

2.1.5.8 addLDAPHost
Online command that adds a new remote host.

**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.

**Syntax**
addLDAPHost(adapterName, host, port, [contextName])

<table>
<thead>
<tr>
<th>Table 2–46  addLDAPHost Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argument</td>
</tr>
<tr>
<td>adapterName</td>
</tr>
<tr>
<td>host</td>
</tr>
<tr>
<td>port</td>
</tr>
<tr>
<td>contextName</td>
</tr>
</tbody>
</table>

**Example**
The following commands add a host and a port to an existing LDAP adapter:
addLDAPHost(adapterName='ldap1', host='myhost.example.com', port=389)
addLDAPHost('ldap1', 'myhost.example.com', '389', 'myContext')

2.1.5.9 addMappingContext
Online command that creates a new mapping context.

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

**Syntax**
addMappingContext(mappingContextId, [contextName])

<table>
<thead>
<tr>
<th>Table 2–47  addMappingContext Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argument</td>
</tr>
<tr>
<td>mappingContextId</td>
</tr>
<tr>
<td>contextName</td>
</tr>
</tbody>
</table>
Example
The following command creates a mapping context for the libOVD configuration:
```
addMappingContext('defaultContext','context')
```

2.1.5.10 addPlugin
Online command that adds a plug-in to an existing adapter or at the global level.

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.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>pluginName</td>
<td>Name of the plug-in to be created.</td>
</tr>
<tr>
<td>pluginClass</td>
<td>Class of the plug-in.</td>
</tr>
<tr>
<td>paramKeys</td>
<td>Init Param Keys separated by &quot;</td>
</tr>
<tr>
<td>paramValues</td>
<td>Init Param Values separated by &quot;</td>
</tr>
<tr>
<td>adapterName</td>
<td>Optional. Name of the adapter to be modified. If not specified, the plug-in is added at the global level.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Examples
The following commands add a plug-in to an existing adapter:
```
wls:/mydomain/serverConfig> addPlugin(adapterName='ldap1', pluginName='VirtualAttr', pluginClass='oracle.ods.virtualization.engine.chain.plugins.virtualattr.VirtualAttributePlugin', paramKeys='AddAttribute | MatchFilter | ContainerDN', paramValues='cn=%uid% | objectclass=person | dc=oracle,dc=com')
wls:/mydomain/serverConfig> addPlugin(pluginName='VirtualAttr', pluginClass='oracle.ods.virtualization.engine.chain.plugins.virtualattr.VirtualAttributePlugin', paramKeys='AddAttribute | MatchFilter | ContainerDN', paramValues='cn=%uid% | objectclass=person | dc=oracle,dc=com')
wls:/mydomain/serverConfig> addPlugin(pluginName='DMSMetrics', pluginClass='oracle.ods.virtualization.engine.chain.plugins.DMSMetrics.MonitorPerformance', paramKeys='None', paramValues='None', adapterName='ldap1', contextName='default')
```

2.1.5.11 addPluginParam
Online command that adds new parameter values to the existing adapter level plug-in or global plug-in.
**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.

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

**Table 2–49  addPluginParam Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>pluginName</td>
<td>Name of the plug-in to be modified.</td>
</tr>
<tr>
<td>paramKeys</td>
<td>Init Param Keys separated by &quot;</td>
</tr>
<tr>
<td>paramValues</td>
<td>Init Param Values separated by &quot;</td>
</tr>
<tr>
<td>adapterName</td>
<td>Optional Name of the adapter to be modified. If not specified, the global plug-in is modified.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

**Examples**
The following commands add a new plug-in parameter for an existing plug-in:

```
wlsec:/mydomain/serverConfig> addPluginParam(adapterName='ldap1',
    pluginName='VirtualAttr', paramKeys='ReplaceAttribute | MatchFilter',
    paramValues='cn=%uid% | objectclass=person')
```

```
wlsec:/mydomain/serverConfig> addPluginParam(pluginName='VirtualAttr',
    paramKeys='ReplaceAttribute | MatchFilter', paramValues='cn=%uid% | objectclass=person')
```

---

**2.1.5.12 addToRequestControlExcludeList**
Online command that adds a control to the Request Control Exclude List.

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

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

**Table 2–50  addToRequestControlExcludeList Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td>control</td>
<td>LDAP control object identifier (OID).</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

**Example**
The following command adds 2.16.840.1.113894.1.8.31 control to ldap1 adapter’s Request Control Exclude List:
addToRequestControlExcludeList (adapterName='ldap1',
control='2.16.840.1.113894.1.8.31', contextName='default')

### 2.1.5.13 addToRequestControlIncludeList

Online command that adds a control to the Request Control Include List.

**Description**

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

**Syntax**

addToRequestControlIncludeList (adapterName, control, [contextName])

**Table 2–51  addToRequestControlIncludeList Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td>control</td>
<td>LDAP control object identifier (OID).</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

**Example**

The following command adds 2.16.840.1.113894.1.8.31 control to ldap1 adapter’s Request Control Include List:

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

### 2.1.5.14 assignViewToAdapter

Online command that assigns a view to an LDAP adapter.

**Description**

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

**Syntax**

assignViewToAdapter (viewName, adapterName, [contextName])

**Table 2–52  assignViewToAdapter Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>viewName</td>
<td>Name of the view.</td>
</tr>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

**Example**

The following command assigns userView to ldap1 adapter:
2.1.5.15 createJoinAdapter
Online command that creates a new join adapter.

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the Join Adapter to be created.</td>
</tr>
<tr>
<td>primaryAdapter</td>
<td>Specifies the identifier of the primary adapter, which is the adapter searched first in the join operation.</td>
</tr>
<tr>
<td>root</td>
<td>root</td>
</tr>
<tr>
<td>bindAdapter</td>
<td>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.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Examples
The following commands create a Join adapter:
createJoinAdapter('join1', 'dc=join', 'primaryldap', 'myldap', 'myContext')
createJoinAdapter(adapterName='join1', root='dc=join', primaryAdapter='myldap')

2.1.5.16 createLDAPAdapter
Online command that creates a new LDAP adapter.

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter to be created.</td>
</tr>
<tr>
<td>root</td>
<td>Virtual Namespace of the LDAP adapter.</td>
</tr>
</tbody>
</table>

assignViewToAdapter('userView', 'ldap1', 'default')
The following commands create an LDAP adapter:

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

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

### 2.1.5.17 createLDAPAdapterWithDefaultPlugins

Online command that creates a new LDAP adapter.

**Description**

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

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>Remote LDAP host with which the LDAP adapter will communicate.</td>
</tr>
<tr>
<td>port</td>
<td>Remote LDAP host port number.</td>
</tr>
<tr>
<td>remoteBase</td>
<td>Location in the remote DIT to which root corresponds.</td>
</tr>
<tr>
<td>isSecure</td>
<td>Optional. Boolean value that enables secure SSL/TLS connections to the</td>
</tr>
<tr>
<td></td>
<td>remote hosts when set to true. The default is false.</td>
</tr>
<tr>
<td>bindDN</td>
<td>Optional. Proxy BindDN used to communicate with remote host. Default is</td>
</tr>
<tr>
<td></td>
<td>&quot;&quot;.</td>
</tr>
<tr>
<td>bindPasswd</td>
<td>Optional. Proxy BindPasswd used to communicate with the remote host.</td>
</tr>
<tr>
<td></td>
<td>Default is &quot;&quot;.</td>
</tr>
<tr>
<td>passCred</td>
<td>Optional. Controls the credentials, if any, the libOVD configuration will</td>
</tr>
<tr>
<td></td>
<td>pass to the back-end (remote host) LDAP server. Values can be Always (</td>
</tr>
<tr>
<td></td>
<td>default), None, or BindOnly.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is</td>
</tr>
<tr>
<td></td>
<td>associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>
Infrastructure Commands

**Examples**

The following commands create an LDAP adapter with default plug-ins based on the directory type:

```bash
wls:/mydomain/serverConfig> createLDAPAdapterWithDefaultPlugins("testLDAP", "OID", "dc=us,dc=oracle,dc=com", "myhost.example.domain.com", 3060, "dc=uk,dc=oid", false, "cn=testuser", "welcome1", "myContext")
```

```bash
wls:/mydomain/serverConfig>
createLDAPAdapterWithDefaultPlugins(adapterName='ldap1', directoryType="OID", root='dc=com', host='myhost.example.domain.com', port=5566, remoteBase='dc=oid',bindDN="cn=testuser",bindPasswd="welcome1",contextName='default')
```

### 2.1.5.18 createView

Online command that creates a new view.

**Description**

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

**Syntax**

```bash
createView(viewName, [contextName])
```
Table 2–56  createView Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>viewName</td>
<td>Name of the new view.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command create a view named, userView:
createView('userView','default')

2.1.5.19 deleteAdapter
Online command that deletes an existing adapter.

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

Syntax
deleteAdapter(adapterName, [contextName])

Table 2–57  deleteAdapter Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the Join Adapter to be deleted.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command deletes join1 adapter:
deleteAdapter(adapterName='join1')
deleteAdapter('join1', 'default'))

2.1.5.20 deleteAttributeExclusionRule
Online command that deletes an attribute exclusion rule.

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

Syntax
deleteAttributeExclusionRule(attribute, mappingContextId, [contextName])

Table 2–58  deleteAttributeExclusionRule Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute</td>
<td>Name of the attribute to be removed from the exclusion list.</td>
</tr>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
</tbody>
</table>
Example
The following command deletes the objectsid attribute exclusion rule for the associated libOVD configuration:
```java
deleteAttributeExclusionRule('objectsid','userrole')
```

### 2.1.5.21 deleteAttributeRule
Online command that delete an attribute mapping rule.

**Description**
Deletes an attribute mapping rule for the libOVD configuration associated with an OPSS context.

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

#### Table 2–59 deleteEntityRelation Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>srcAttrs</code></td>
<td>Source attributes.</td>
</tr>
<tr>
<td><code>dstAttr</code></td>
<td>Destination attribute.</td>
</tr>
<tr>
<td><code>mappingContextId</code></td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td><code>contextName</code></td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command deletes the lastname attribute mapping rule from cn:
```java
deleteAttributeRule('lastname','sn')
```

### 2.1.5.22 deleteDomainExclusionRule
Online command that deletes a domain exclusion rule.

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

**Syntax**
```java
deleteDomainExclusionRule(domain, mappingContextId, [contextName])
```
### Table 2–60 deleteEntityRelation Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>Distinguished Name of the container to be removed from the exclusion list.</td>
</tr>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

#### Example
The following command deletes 'cn=group,dc=oracle,dc=com' domain exclusion rule:

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

#### 2.1.5.23 deleteDomainRule
Online command that deletes a domain mapping rule.

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

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

### Table 2–61 deleteDomainRule Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>srcDomain</td>
<td>Source domain.</td>
</tr>
<tr>
<td>destDomain</td>
<td>Destination domain.</td>
</tr>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

#### Example
The following command deletes 'dc=oracle,dc=com' domain mapping rule:

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

#### 2.1.5.24 deleteDomainExclusionRule
Deletes a domain exclusion rule.

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

**Syntax**
```
deleteDomainExclusionRule(domain, mappingContextId, [contextName])
```
Table 2–62  deleteDomainExclusionRule Attributes

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>Distinguished Name of the container to be removed from the exclusion list.</td>
</tr>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default value is default.</td>
</tr>
</tbody>
</table>

Example

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

2.1.5.25 deleteMappingContext

Online command that deletes a mapping context.

Description

Deletes the specified mapping context for the libOVD configuration associated with an OPSS context.

Syntax

deleteMappingContext {mappingContextId, [contextName]}

Table 2–63  deleteMappingContext Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example

The following command deletes a mapping context for a libOVD configuration:

deleteMappingContext {'defaultContext','context'}

2.1.5.26 deleteView

Online command that deletes a view.

Description

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

Syntax

ccreateView {viewName, [contextName]}

Table 2–64  createView Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>viewName</td>
<td>Name of the view to delete.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>
Example
The following command deletes userView view:

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

2.1.5.27 getAdapterDetails
Online command that displays the details of an existing adapter.

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the adapter that contains the details to be displayed.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Examples
The following commands display the details of ldap1 and join1 adapter respectively:
```
getAdapterDetails(adapterName='ldap1', contextName='default')
getAdapterDetails(adapterName='join1')
```

2.1.5.28 listAdapters
Online command that lists the name and type of all adapters.

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

Syntax
```
listAdapters([contextName])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command displays the name and type of all adapters configured for a libOVD configuration:
```
listAdapters()
listAdapters(contextName='myContext')
```
### 2.1.5.29 listAllMappingContextIds

Online command that lists all mapping contexts.

**Description**

Lists the mapping contexts associated with the specified OPSS context.

**Syntax**

```
listAllMappingContextIds([contextName])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

**Example**

The following command lists all the mapping contexts:

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

### 2.1.5.30 listAttributeRules

Online command that lists all the attribute rules.

**Description**

List all the attribute rules in the format `SOURCE_ATTRIBUTE:DESTINATION_ATTRIBUTE:DIRECTION`.

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

**Example**

The following command lists all the attribute rules:

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

### 2.1.5.31 listDomainRules

Online command that lists all domain rules.

**Description**

Lists all the domain rules in the format of `SOURCE_DOMAIN:DESTINATION_DOMAIN`.

**Syntax**

```
listDomainRules(mappingContextId, [contextName])
```
### Table 2–69  listDomainRules Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>mappingContextId</td>
<td>Name of the mapping context.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

**Example**
The following command lists all domain rules:

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

### 2.1.5.32 listViews

**Online command that lists all views**

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

**Syntax**

```
listViews([contextName])
```

### Table 2–70  listViews Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

**Example**
The following command lists all views:

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

### 2.1.5.33 modifyLDAPAdapter

**Modifies parameters in an LDAP adapter.**

**Description**
Modifies the following LDAP adapter parameters:

- Remote Base
- Root
- Secure
- BindDN
- BindPassword
- PassCredentials
- MaxPoolSize
- MaxPoolChecks
- MaxPoolWait
- InitialPoolSize
- PoolCleanupInterval
- MaxPoolConnectionIdleTime
- Active
- PingProtocol
- PingBindDN
- PingBindPassword
- PageSize
- HeartBeatInterval
- OperationTimeout
- SearchCountLimit
- Visible
- Critical
- InclusionFilter
- ExclusionFilter
- DNPattern
- RequestControlAllowServerSupported
- MaxPoolConnectionReuseTime
- ConnectTimeout
- PoolConnectionReclaimTime
- Protocols

**Syntax**

```plaintext
modifyLDAPAdapter(adapterName, attribute, value, [contextName])
```

**Table 2–71 modifyLDAPAdapter Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>adapterName</code></td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td><code>attribute</code></td>
<td>Name of the attribute to be modified.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>New value for the attribute.</td>
</tr>
<tr>
<td><code>contextName</code></td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

**Examples**

The following examples illustrate how to set attributes in `ldap1`:

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

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

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

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

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

modifyLDAPAdapter (adapterName='ldap1', attribute='MaxPoolConnectionReuseTime', value=3600, contextName='mydefault') [value is in seconds]

modifyLDAPAdapter (adapterName='ldap1', attribute='ConnectTimeout', value=10000, contextName='mydefault') [value is in milli seconds]

modifyLDAPAdapter (adapterName='ldap1', attribute='PoolConnectionReclaimTime', value=180, contextName='mydefault')

modifyLDAPAdapter (adapterName='ldap1', attribute='Protocols', value='TLSv1.2', contextName='mydefault')

2.1.5.34 modifySocketOptions
Online command that modifies socket options.

Description
Modifies socket options for an existing LDAP adapter configuration.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td>reuseAddress</td>
<td>Value of reuseAddress.</td>
</tr>
<tr>
<td>keepAlive</td>
<td>Value of keepAlive.</td>
</tr>
<tr>
<td>tcpNoDelay</td>
<td>Value of tcpNoDelay.</td>
</tr>
<tr>
<td>readTimeout</td>
<td>Value of readTimeout in seconds.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command modifies the socket option for ldap1 adapter:

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

2.1.5.35 removeAllRequestControlExcludeList
Online command that removes all controls from the Request Control Exclude List.
Description
Removes all controls from the Request Control Exclude List for an existing LDAP adapter configuration.

Syntax
removeAllRequestControlExcludeList(adapterName, [contextName])

Table 2–73  removeAllRequestControlExcludeList Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the adapter to be updated.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command removes all controls from ldap1 adapter’s Request Control Exclude List:
removeAllRequestControlExcludeList(adapterName='ldap1', contextName='default')

2.1.5.36 removeAllRequestControlIncludeList
Online command that removes all controls from the Request Control Include List.

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

Syntax
removeAllRequestControlIncludeList(adapterName, [contextName])

Table 2–74  removeAllRequestControlIncludeList Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the adapter to be updated.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command removes all controls from ldap1 adapter’s Request Control Include List:
removeAllRequestControlIncludeList(adapterName='ldap1', contextName='default')

2.1.5.37 removeFromRequestControlExcludeList
Online command that removes a control from the Request Control Exclude List.

Description
Removes a control from the Request Control Exclude List for an existing LDAP adapter configuration.
Syntax
removeFromRequestControlExcludeList(adapterName, control, [contextName])

Table 2–75  removeFromRequestControlExcludeList Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td>control</td>
<td>LDAP control object identifier (OID).</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command removes 2.16.840.1.113894.1.8.31 control from ldap1 adapter’s Request Control Exclude List:
removeFromRequestControlExcludeList({adapterName='ldap1', control='2.16.840.1.113894.1.8.31', contextName='default'})

2.1.5.38 removeDNAttribute
Online command that removes a attribute from the DN Attributes List.

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

Syntax
removeDNAttribute(adapterName attributeName, [contextName])

Table 2–76  removeDNAttribute Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the adapter to be updated.</td>
</tr>
<tr>
<td>attributeName</td>
<td>Name of the new DN attribute to be removed.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command removes memberof attribute from ldap1 adapter’s attribute list:
removeDNAttribute({adapterName='ldap1', attributeName='memberof', contextName='default'})

2.1.5.39 removeFromRequestControlIncludeList
Online command that removes a control from the Request Control Include List.

Description
Removes a control from the Request Control Include List for an existing LDAP adapter configuration.
Syntax
removeFromRequestControlIncludeList(adapterName, control, [contextName])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td>control</td>
<td>LDAP control object identifier (OID).</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command removes 2.16.840.1.113894.1.8.31 control from ldap1 adapter’s Request Control Include List:
removeFromRequestControlIncludeList(adapterName='ldap1', control='2.16.840.1.113894.1.8.31', contextName='default')

2.1.5.40 removeJoinRule
Online command that removes a join rule from a Join Adapter.

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

Syntax
removeJoinRule(adapterName, secondary, [contextName])

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the Join Adapter to be modified.</td>
</tr>
<tr>
<td>secondary</td>
<td>The join rules corresponding to this secondary adapter are removed from the Join Adapter.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Examples
The following command removes 2.16.840.1.113894.1.8.31 control from ldap1 adapter’s Request Control Include List:
removeJoinRule('join1','secondaryldap1', 'default')
removeJoinRule(adapterName='join1', secondary='LDAP3')

2.1.5.41 removeLDAPHost
Online command that removes a remote host from an existing LDAP adapter.

Description
Removes a remote host (host:port) from an existing LDAP adapter.
Syntax
removeLDAPHost(adapterName, host, [contextName])

Table 2–79  removeLDAPHost Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td>host</td>
<td>Location of a remote LDAP host with which the LDAP adapter will communicate.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command removes the host and port from ldap1 adapter:
removeLDAPHost(adapterName='ldap1', host='myhost.example.com')
removeLDAPHost('ldap1', 'myhost.example.com', 'myContext')

2.1.5.42 removePlugin
Online command that removes a plug-in from an existing adapter.

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

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

Table 2–80  removePlugin Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>pluginName</td>
<td>Name of the plug-in to be removed.</td>
</tr>
<tr>
<td>adapterName</td>
<td>Optional. Name of the adapter to be modified. If not specified, the global plug-in is removed.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following commands remove VirtualAttr plug-in from ldap1 adapter:
removePlugin(adapterName='ldap1', pluginName='VirtualAttr')
removePlugin(pluginName='VirtualAttr')

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

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

**Table 2–81  removePluginParam Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pluginName</td>
<td>Name of the plug-in to be modified.</td>
</tr>
<tr>
<td>paramKey</td>
<td>Parameter to be removed.</td>
</tr>
<tr>
<td>adapterName</td>
<td>Optional. Name of the adapter to be modified. If not specified, the global plug-in is modified.</td>
</tr>
</tbody>
</table>
| contextName| Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is “default”.

**Example**
The following commands remove the plug-in parameter ReplaceAttribute from VirtualAttr plug-in:
```
removePluginParam(adapterName='ldap1', pluginName='VirtualAttr', paramKey='ReplaceAttribute')
removePluginParam(pluginName='VirtualAttr', paramKey='ReplaceAttribute')
```

2.1.5.44 replacePluginParam

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

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

**Table 2–82  replacePluginParam Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pluginName</td>
<td>Name of the plug-in to be modified.</td>
</tr>
<tr>
<td>paramName</td>
<td>Name of the parameter to be replaced.</td>
</tr>
<tr>
<td>paramValues</td>
<td>New values of the parameter. For more than one new value, separate each new parameter value by a &quot;</td>
</tr>
<tr>
<td>adapterName</td>
<td>Optional. Name of the adapter to be modified. If not specified, the global plug-in is modified.</td>
</tr>
</tbody>
</table>
| contextName| Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is “default”.

Examples
The following commands replace the parameter values for the associated plug-in for an adapter:

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

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

2.1.5.45 unassignViewFromAdapter
Online command that unassigns a view from an adapter.

Description
Unassigns a view from an LDAP adapter configuration.

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

Table 2–83  unassignViewFromAdapter Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>viewName</td>
<td>Name of the view.</td>
</tr>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which the libOVD configuration is associated. Default is &quot;default&quot;.</td>
</tr>
</tbody>
</table>

Example
The following command unassigns userView associated with ldap1 adapter:
```
unassignViewFromAdapter('userView','ldap1', 'default')
```

2.1.5.46 listSSLStoreType
Online command that lists the type of SSL store in use.

Description
This command lists the type of SSL store in use for libOVD (JKS or KSS).

Syntax
```
listSSLStoreType(contextName=[contextName])
```

Table 2–84  listSSLStoreType Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextName</td>
<td>Name of the OPSS context with which libOVD configuration is associated. The default value is default.</td>
</tr>
</tbody>
</table>

Example
This following command list the SSL store types in use:
```
listSSLStoreType(contextName='default')
```
2.1.5.47 enableKSSForSSL
Online command to enable KSS for libOVD.

**Description**
This command enables KSS for SSL, and disables JKS if it was enabled before. For more information about KSS, see Oracle® Fusion Middleware Securing Applications with Oracle Platform Security Services.

**Syntax**
`enableKSSForSSL(contextName=[contextName])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which libOVD configuration is associated. The default value is default.</td>
</tr>
</tbody>
</table>

**Example**
The following command enables KSS for SSL:

```shell
enableKSSForSSL(contextName='default')
```

2.1.5.48 enableJKSForSSL
Online command to enable JKS for libOVD.

**Description**
This command enables JKS for SSL, and disables KSS if it was enabled before. The command assumes that the libOVD adapters.jks file exists.

**Syntax**
`enableJKSForSSL(contextName=[contextName])`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which libOVD configuration is associated. The default value is default.</td>
</tr>
</tbody>
</table>

**Example**
The following command enables JKS for SSL:

```shell
enableJKSForSSL(contextName='default')
```

2.1.5.49 createKeyStoreAndEnableJKSForSSL
Online command to enable JKS for SSL.

**Description**
This command enables JKS for SSL, and disables KSS if it was enabled before. The command creates the libOVD adapters.jks file.
**Infrastructure Commands**

**Syntax**
```
createKeyStoreAndEnableJKSforSSL(keystorePassword=[password],
contextName=[contextName])
```

**Table 2–87 createKeyStoreAndEnableJKSforSSL Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>keystorePassword</td>
<td>Password for libOVD adapters.jks file.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which libOVD configuration is associated. The default value is default.</td>
</tr>
</tbody>
</table>

**Example**
The following command enable JKS for SSL:
```
createKeyStoreAndEnableJKSforSSL(keystorePassword='welcome1',
contextName='default')
```

### 2.1.5.50 importTrustedCertificateIntoSSLStore

Online command to import trusted certificate into SSL store.

**Description**
This command imports the provided trusted certificate into SSL store.

**Syntax**
```
importTrustedCertificateIntoSSLStore(certificateFileName=[cert_file],
aliasName=[aliasName],contextName=[contextName])
```

**Table 2–88 importTrustedCertificateIntoSSLStore Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificateFileName</td>
<td>File name that contains the certificate.</td>
</tr>
<tr>
<td>aliasName</td>
<td>Alias name for the certificate.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which libOVD configuration is associated. The default value is default.</td>
</tr>
</tbody>
</table>

**Example**
The following command imports the provided trusted certificate into SSL store:
```
importTrustedCertificateIntoSSLStore(certificateFileName='/tmp/cert.txt',aliasName='myCert1',contextName='default')
```

### 2.1.5.51 migrateAllTrustedCertificatesFromJKSToKSS

Online command to migrate all trusted certificates from JKS-based libOVD truststore to KSS store.

**Description**
This command migrates all trusted certificates from JKS-based libOVD truststore to KSS store.
Syntax
migrateAllTrustedCertificatesFromJKSToKSS(contextName=[contextName])

Table 2–89  migrateAllTrustedCertificatesFromJKSToKSS Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which libOVD configuration is associated. The default value is default.</td>
</tr>
</tbody>
</table>

Example
The following command migrates all trusted certificates from JKS-based libOVD truststore to KSS store:
migrateAllTrustedCertificatesFromJKSToKSS(contextName='default')

2.1.5.52  migrateTrustedCertificatesFromJKSToKSS
Online command to migrate given trusted certificates from JKS-based libOVD truststore to KSS store.

Description
This command migrates the given trusted certificates from JKS-based libOVD truststore to KSS store.

Syntax
migrateTrustedCertificatesFromJKSToKSS(aliasNames=[alias_names], contextName=[contextName])

Table 2–90  migrateTrustedCertificatesFromJKSToKSS Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>aliasNames</td>
<td>List of alias names to migrate separated by a comma.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which libOVD configuration is associated. The default value is default.</td>
</tr>
</tbody>
</table>

Example
The following command migrates the specified trusted certificates from JKS-based libOVD truststore to KSS store:
migrateTrustedCertificatesFromJKSToKSS (aliasNames='alias1,alias2', contextName='default')

2.1.5.53  changeLDAPHostPort
Online command to change given LDAP host and port in an existing LDAP adapter configuration to a new host and port.

Description
This command changes given LDAP host and port in an existing LDAP adapter configuration to a new host and port.
**Syntax**

```
changeLDAPHostPort(adapterName=[adapterName], oldHost=[oldHost],
oldPort=[oldPort], newHost=[newHost], newPort=[newPort],
contextName=[contextName])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>adapterName</code></td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td><code>oldHost</code></td>
<td>Old LDAP host.</td>
</tr>
<tr>
<td><code>oldPort</code></td>
<td>Old LDAP port.</td>
</tr>
<tr>
<td><code>newHost</code></td>
<td>New LDAP host.</td>
</tr>
<tr>
<td><code>newPort</code></td>
<td>New LDAP port.</td>
</tr>
<tr>
<td><code>contextName</code></td>
<td>Optional. Name of the OPSS context with which libOVD configuration is associated. The default value is <code>default</code>.</td>
</tr>
</tbody>
</table>

**Example**

The following command changes given LDAP host and port in an existing LDAP adapter configuration to a new host and port:

```
changeLDAPHostPort(adapterName='ldap1', oldHost='oldhost.example.domain.com',
oldPort=389, newHost='newhost.example.domain.com', newPort=389)
```

### 2.1.5.54 removeLDAPHostPort

Online command to remove a remote host and a port from an existing LDAP adapter configuration.

**Description**

This command removes a remote host and a port from an existing LDAP adapter configuration.

**Syntax**

```
removeLDAPHostPort(adapterName=[adapterName], host=[host], port=[port],
contextName=[contextName])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>adapterName</code></td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td><code>host</code></td>
<td>Remote LDAP host.</td>
</tr>
<tr>
<td><code>port</code></td>
<td>Remote LDAP port.</td>
</tr>
<tr>
<td><code>contextName</code></td>
<td>Optional. Name of the OPSS context with which libOVD configuration is associated. The default value is <code>default</code>.</td>
</tr>
</tbody>
</table>

**Example**

The following command removes a remote host and a port from an existing LDAP adapter configuration:

```
removeLDAPHostPort(adapterName='ldap1', host='myhost.example.domain.com',
port=389)
```
2.1.5.55 setReadOnlyForLDAPHost

Online command to set the given host and port to read-only/writable in an existing LDAP adapter configuration.

**Description**

This command sets the given host and port to read-only/writable in an existing LDAP adapter configuration.

**Syntax**

```
setReadOnlyForLDAPHost(adapterName=[adapterName], host=[host], port=[port], readOnly=[true/false], contextName=[contextName])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter to be modified.</td>
</tr>
<tr>
<td>host</td>
<td>LDAP host.</td>
</tr>
<tr>
<td>port</td>
<td>LDAP port.</td>
</tr>
<tr>
<td>readOnly</td>
<td>It has values: true or false.</td>
</tr>
<tr>
<td>contextName</td>
<td>Optional. Name of the OPSS context with which libOVD configuration is associated. The default value is default.</td>
</tr>
</tbody>
</table>

**Example**

The following command sets the given host and port to read-only in an existing LDAP adapter configuration:

```
setReadOnlyForLDAPHost(adapterName='ldap1', host='myhost.example.domain.com', port=389, readOnly=true)
```

2.1.5.56 dumpLdapConnectionPoolStats

Online command that dumps the current connection pool statistics for an adapter to a file for the given JVM on which WLS is configured.

**Description**

This command dumps the current connection pool statistics for an adapter to a file for the given JVM on which WLS is configured.

**Syntax**

```
dumpLdapConnectionPoolStats(fileName=[fileName], adapterName=[adapterName], contextName=[contextName])
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileName</td>
<td>Refers to the full path of the file.</td>
</tr>
<tr>
<td>adapterName</td>
<td>Name of the LDAP adapter.</td>
</tr>
</tbody>
</table>
\textbf{Example}

The following example dumps the connection pool statistics for \textit{ldap1} adapter into the specified file:

\begin{verbatim}
dumpLdapConnectionPoolStats('/tmp/poolstats1.txt','ldap1', 'default')
\end{verbatim}
This chapter describes SSL configuration WLST commands.
This chapter contains the following sections:

- About SSL Configuration Commands
- Properties Files for SSL

### 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>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>configureSSL</td>
<td>Set the SSL attributes for a component listener.</td>
<td>Online</td>
</tr>
<tr>
<td>getSSL</td>
<td>Display the SSL attributes for a component listener.</td>
<td>Online</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Key</th>
<th>Mandatory?</th>
<th>Allowed Values for Oracle HTTP Server</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSLEnabled</td>
<td>No</td>
<td>true, false</td>
<td>Either value</td>
</tr>
<tr>
<td>Ciphers</td>
<td>No</td>
<td>SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA, SSL_RSA_WITH_3DES_EDE_CBC_SHA, SSL_RSA_WITH_DES_CBC_SHA, SSL_DH_anon_WITH_RC4_128_MD5, SSL_DH_anon_WITH_DES_CBC_SHA, SSL_DH_anon_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA</td>
<td>One or more comma-separated values</td>
</tr>
<tr>
<td>SSLVersions</td>
<td>No</td>
<td>nzos_Version_3_0, nzos_Version_3_0_With_2_0_Hello, nzos_Version_1_0</td>
<td>One or more comma-separated values</td>
</tr>
<tr>
<td>CertValidation</td>
<td>No</td>
<td>none, crl</td>
<td>Either value</td>
</tr>
<tr>
<td>CertValidationPath</td>
<td>No</td>
<td>file://crl_file_path, dir://crl_dir_path</td>
<td>Path of the CRL file, or directory containing CRL files</td>
</tr>
</tbody>
</table>

Table 3–3 shows the default values:
Table 3–3  Default Values of Parameters

<table>
<thead>
<tr>
<th>Key</th>
<th>Default Value for Oracle HTTP Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSLEnabled</td>
<td>true</td>
</tr>
<tr>
<td>Ciphers</td>
<td>null</td>
</tr>
<tr>
<td>SSLVersions</td>
<td>null</td>
</tr>
<tr>
<td>CertValidation</td>
<td>none</td>
</tr>
<tr>
<td>CertValidationPath</td>
<td>null</td>
</tr>
<tr>
<td>KeyStore</td>
<td>default</td>
</tr>
<tr>
<td>TrustStore</td>
<td>-</td>
</tr>
<tr>
<td>AuthenticationType</td>
<td>Server</td>
</tr>
</tbody>
</table>

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 nzsos_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:
  
  $\text{SSLEnabled}=\text{true}$

  Here the value $\text{true}$ is explicitly specified.

- The setting:
  
  $\text{SSLEnabled}=\text{}$

  Since no value is mentioned here, the default value of $\text{SSLEnabled}$ ($\text{true}$) is used.

- The key $\text{SSLEnabled}$ is not present in the properties file.

  Since the key is not present, its default value ($\text{true}$) is used.

**Example 3: Properties File with Version for OHS**

```plaintext
SSLEnabled=true
AuthenticationType=Mutual
SSLVersion=nzos_Version_3_0
CertValidation=crl
CertValidationPath=file:///tmp/file.crl
KeyStore=ohs1
```

This properties file has:

- Default values for ciphers
- Keystore
- SSL version v3
- CRL validation turned on
- Mutual Authentication mode
configureSSL

Online command that sets SSL attributes.

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".

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>listener</td>
<td>Specifies the name of the component listener to be configured for SSL.</td>
</tr>
<tr>
<td>filePath</td>
<td>Specifies the absolute path of the properties file containing the SSL attributes to set.</td>
</tr>
</tbody>
</table>

Example

Here are some examples of configureSSL command usage.

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')
getSSL

Online command that lists the configured SSL attributes.

Description

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

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>listener</td>
<td>Specifies the name of the component listener.</td>
</tr>
</tbody>
</table>

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')
This chapter describes how to configure Oracle wallets using WLST commands.
This chapter contains the following topic:

- The WLST Wallet Commands

### 4.1 The WLST Wallet Commands

WLST commands allow to manage Oracle wallets for Oracle Fusion Middleware components. Table 4–1 lists the available commands.

To obtain help for a command, invoke a command like the following:

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

Certain commands require parameters like instance name, ias-component or process type. To obtain such information, invoke commands like the following:

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

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

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>addCertificateRequest</td>
<td>Generate a certificate signing request in an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>addSelfSignedCertificate</td>
<td>Add a self-signed certificate to an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>changeWalletPassword</td>
<td>Change the password to an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>createWallet</td>
<td>Create an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>deleteWallet</td>
<td>Delete an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>exportWallet</td>
<td>Export an Oracle wallet to a file.</td>
<td>Online</td>
</tr>
<tr>
<td>exportWalletObject</td>
<td>Export an object (for example, a certificate) from an Oracle wallet to a file.</td>
<td>Online</td>
</tr>
<tr>
<td>getWalletObject</td>
<td>Display a certificate or other object present in an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>importWallet</td>
<td>Import an Oracle wallet from a file.</td>
<td>Online</td>
</tr>
</tbody>
</table>
### Table 4–1 (Cont.) WLST Commands for Oracle Wallet Management

<table>
<thead>
<tr>
<th>Use this command...</th>
<th>To...</th>
<th>Use with WLST...</th>
</tr>
</thead>
<tbody>
<tr>
<td>importWalletObject</td>
<td>Import a certificate or other object from a file to an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>listWalletObjects</td>
<td>List all objects (such as certificates) present in an Oracle wallet.</td>
<td>Online</td>
</tr>
<tr>
<td>listWallets</td>
<td>List all Oracle wallets configured for a component instance.</td>
<td>Online</td>
</tr>
<tr>
<td>removeWalletObject</td>
<td>Remove a certificate or other object from a component instance's Oracle wallet.</td>
<td>Online</td>
</tr>
</tbody>
</table>

**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.
addCertificateRequest

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

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.

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>DN</td>
<td>Specifies the Distinguished Name of the key pair entry.</td>
</tr>
<tr>
<td>keySize</td>
<td>Specifies the key size in bits.</td>
</tr>
</tbody>
</table>

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',)
```
addSelfSignedCertificate

Online command that adds a self-signed certificate.

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.

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>DN</td>
<td>Specifies the Distinguished Name of the key pair entry.</td>
</tr>
<tr>
<td>keySize</td>
<td>Specifies the key size in bits.</td>
</tr>
</tbody>
</table>

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')
changeWalletPassword

Online command that changes the password of an Oracle wallet.

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.

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the filename of the wallet.</td>
</tr>
<tr>
<td>currPassword</td>
<td>Specifies the current wallet password.</td>
</tr>
<tr>
<td>newPassword</td>
<td>Specifies the new wallet password.</td>
</tr>
</tbody>
</table>

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')
createWallet

Online command that creates an Oracle wallet.

**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.

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file to be created.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the wallet password.</td>
</tr>
</tbody>
</table>

**Example**

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', '')
```
deleteWallet

Online command that deletes an Oracle wallet.

Description

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

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file to be deleted.</td>
</tr>
</tbody>
</table>

Example

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

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

Online command that exports an Oracle wallet.

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.

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>path</td>
<td>Specifies the absolute path of the directory under which the object is exported.</td>
</tr>
</tbody>
</table>

Example

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')
exportWalletObject

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

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.

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of wallet object to be exported. Valid values are 'CertificateRequest', 'Certificate', 'TrustedCertificate' or 'TrustedChain'.</td>
</tr>
<tr>
<td>path</td>
<td>Specifies the absolute path of the directory under which the object is exported as a file base64.txt.</td>
</tr>
<tr>
<td>DN</td>
<td>Specifies the Distinguished Name of the wallet object being exported.</td>
</tr>
</tbody>
</table>

Example

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`:

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

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

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.

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of wallet object to be exported. Valid values are 'CertificateRequest', 'Certificate', and 'TrustedCertificate'.</td>
</tr>
<tr>
<td>index</td>
<td>Specifies the index number of the wallet object as returned by the listWalletObjects command.</td>
</tr>
</tbody>
</table>

Example

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:

```bash
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:

```bash
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 server instance inst1:

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

Online command that imports an Oracle wallet from a file.

**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.

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>(walletName)</td>
<td>Specifies the name of the wallet being imported. The name must be unique for the component instance.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>filePath</td>
<td>Specifies the absolute path of the wallet file being imported.</td>
</tr>
</tbody>
</table>

**Example**

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:

```bash
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:

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

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

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.

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of wallet object to be imported. Valid values are 'Certificate', 'TrustedCertificate' and 'TrustedChain'.</td>
</tr>
<tr>
<td>filePath</td>
<td>Specifies the absolute path of the file containing the wallet object.</td>
</tr>
</tbody>
</table>

Example

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')
```
listWalletObjects

Online command that lists all objects in an Oracle wallet.

Description

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

Syntax

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of wallet object to be listed. Valid values are 'CertificateRequest', 'Certificate', and 'TrustedCertificate'.</td>
</tr>
</tbody>
</table>

Example

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')
```
**listWallets**

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

**Description**

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

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
</tbody>
</table>

**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')
```
**removeWalletObject**

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

**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.

**Syntax**

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>instName</td>
<td>Specifies the name of the application server instance.</td>
</tr>
<tr>
<td>compName</td>
<td>Specifies the name of the component instance.</td>
</tr>
<tr>
<td>compType</td>
<td>Specifies the type of component. Valid value is 'ohs'.</td>
</tr>
<tr>
<td>walletName</td>
<td>Specifies the name of the wallet file.</td>
</tr>
<tr>
<td>password</td>
<td>Specifies the password of the wallet.</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the type of the keystore object to be removed. Valid values are 'CertificateRequest', 'Certificate', 'TrustedCertificate' or 'TrustedAll'.</td>
</tr>
<tr>
<td>DN</td>
<td>Specifies the Distinguished Name of the wallet object to be removed.</td>
</tr>
</tbody>
</table>

**Example**

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')
```