Oracle® Fusion Middleware

WLST Command Reference for Infrastructure Security

12*c* (12.2.1.4.0)
F30123-05
December 2023

Oracle Fusion Middleware WLST Command Reference for Infrastructure Security, 12*c* (12.2.1.4.0)

F30123-05

Contents

3 Audit Configuration WLST Commands

4 OPSS Keystore Service Commands

5 SSL Configuration WLST Commands

6 Wallet Configuration WLST Commands

List of Tables

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.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documentation

Additional information is found in the following documents:

- *Securing Applications with Oracle Platform Security Services*
- *Administering Oracle Fusion Middleware*
- *Administering 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:

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

1

Introduction and Roadmap

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

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.

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.
- OPSS Security Store WLST Commands provides detailed descriptions of the custom WLST commands for OPSS security store.
- Audit Configuration WLST Commands provides detailed descriptions of WLST commands for audit configuration.
- OPSS Keystore Service Commands provides detailed descriptions of WLST commands used with the OPSS keystore service.
- SSL Configuration WLST Commands provides detailed descriptions for the SSL configuration WLST commands.
- Wallet Configuration WLST Commands provides detailed descriptions of the WLST commands that you can use to configure Oracle wallets.

2
OPSS Security Store WLST Commands

This chapter describes the OPSS security store commands.
Use the WLST security commands listed in Table 2-1 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 2-1 WLST Security Commands

Use this command...	To...	Use with WLST...
addBootStrapCredential	Add a credential to the bootstrap credential store	Offline
addResourceToEntitlement	Add a resource to an entitlement.	Online
createAppRole	Create a new application role.	Online
createCred	Create a new credential.	Online
createEntitlement	Create an entitlement.	Online
createResource	Create a resource.	Online
createResourceType	Create a new resource type.	Online
deleteAppPolicies	Remove all policies in an application.	Online
deleteAppRole	Remove an application role.	Online
deleteCred	Remove a credential.	Online
deleteEntitlement	Remove an entitlement.	Online
deleteResource	Remove a resource.	Online
deleteResourceType	Remove an existing resource type.	Online
exportEncryptionKey	Export the domain encryption key to the file `ewallet.p12`.	Offline
getEntitlement	List an entitlement.	Online
getResourceType	Fetch an existing resource type.	Online
grantAppRole	Add a principal to a role.	Online
grantEntitlement	Create an entitlement.	Online
grantPermission	Create a new permission.	Online
importEncryptionKey	Import the encryption key in file `ewallet.p12` to the domain.	Offline
listAppRoles	List all roles in an application.	Online
listAppRolesMembers	List all members in an application role.	Online
listAppStripes	List application stripes in policy store.	Online
listCodeSourcePermissions	List permissions assigned to a source code in global policies.	Online
listEntitlement	List an entitlement.	Online
listEntitlements	List entitlements in an application stripe.	Online

Table 2-1 (Cont.) WLST Security Commands

Use this command...	To...	Use with WLST...
listPermissions	List all permissions granted to a principal.	Online
listResourceActions	List actions in a resource.	Online
listResources	List resources in an application stripe.	Online
listResourceTypes	List resource types in an application stripe.	Online
listSecurityStoreInfo	List the type and location of the OPSS security store, and the user allowed to access it.	Offline
migrateSecurityStore	Migrate policies or credentials from a source repository to a target repository.	Offline
modifyBootStrapCredential	Update bootstrap credential store	Offline
reassociateSecurityStore	Reassociate policies and credentials to an LDAP repository	Online
restoreEncryptionKey	Restore the domain encryption key as it was before the last importing.	Offline
revokeAppRole	Remove a principal from a role.	Online
revokeEntitlement	Remove an entitlement.	Online
revokePermission	Remove a permission.	Online
revokeResourceFromEntitlement	Remove a resource from an entitlement	Online
rollOverEncryptionKey	Replace the current domain encryption key with a new one.	Offline
updateCred	Modify the attribute values of a credential.	Online
updateTrustServiceConfig	Update the configuration of the trust service.	Online

Note:

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

- addBootStrapCredential
- addResourceToEntitlement
- createAppRole
- createCred
- createEntitlement
- createResource
- createResourceType
- deleteAppPolicies
- deleteAppRole
- deleteCred

- deleteEntitlement
- deleteResource
- deleteResourceType
- exportEncryptionKey
- getEntitlement
- getResourceType
- grantAppRole
- grantEntitlement
- grantPermission
- importEncryptionKey
- listAppRoles
- listAppRolesMembers
- listAppStripes
- listCodeSourcePermissions
- listEntitlement
- listEntitlements
- listPermissions
- listResourceActions
- listResources
- listResourceTypes
- listSecurityStoreInfo
- migrateSecurityStore
- modifyBootStrapCredential
- reassociateSecurityStore
- restoreEncryptionKey
- revokeAppRole
- revokeEntitlement
- revokePermission
- revokeResourceFromEntitlement
- rollOverEncryptionKey
- updateCred
- updateTrustServiceConfig

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	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run. **Note:** You can specify a `jps-config.xml` file or a `jps-config-jse.xml` file for the `jpsConfigFile` argument.
map	Specifies the map of the credential to add.
key	Specifies the key of the credential to add.
username	Specifies the name of the user in the credential to add.
password	Specifies the password of the user in the credential to add.

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='password')
```

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")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is located.
name	Specifies the name of the entitlement to modify.
resourceName	Specifies the name of the resource to add.
resourceType	Specifies the type of the resource to add. The passed resource type *must* be present in the application stripe at the time this script is invoked.
actions	Specifies the comma-separated list of actions for the added resource.

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")
```

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)
```

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

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")
```

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])
```

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

Example

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

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

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"])
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is created.
name	Specifies the name of the entitlement created.

Argument	Definition
resourceName	Specifies the name of the one resource member of the entitlement created.
actions	Specifies a comma-separated the list of actions for the resource resourceName.
displayName	Specifies the display name of the resource created. Optional.
description	Specifies the description of the entitlement created. Optional.

Example

The following example creates the entitlement myEntitlement with just the resource myResource in the stripe myApplication:

```
wls:/mydomain/serverConfig> createEntitlement(appStripe="myApplication",
name="myEntitlement", resourceName="myResource", actions="read,write")
```

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"])
```

Argument	Definition
appStripe	Specifies the application stripe where the resource is created.
name	Specifies the name of the resource created.
type	Specifies the type of resource created. The passed resource type *must* be present in the application stripe at the time this script is invoked.
displayName	Specifies the display name of the resource created. Optional.
description	Specifies the description of the resource created. Optional.

Example

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

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

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 [, delimeter])
```

Argument	Definition
appStripe	Specifies the application stripe where to insert the resource type.
resourceTypeName	Specifies the name of the resource type to insert.
displayName	Specifies the name for the resource type used in UI gadgets.
`description`	Specifies a brief description of the resource type.
provider	Specifies the provider for the resource type.
matchere	Specifies the class of the resource type. If unspecified, it defaults to `oracle.security.jps.ResourcePermission`.
actions	Specifies the actions allowed on instances of the resource type.
delimeter	Specifies the character used to delimit the list of actions. If unspecified, it defaults to comma ','.

Example

The following example creates a resource type in the stripe myApplication with actions BWPrint and ColorPrint delimited by a semicolon:

```
wls:/mydomain/serverConfig> createResourceType(appStripe="myApplication",
resourceTypeName="resTypeName", displayName="displName", description="A resource
type", provider="Printer", matcher="com.printer.Printer",
actions="BWPrint;ColorPrint" [, delimeter=";"])
```

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)
```

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

Example

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

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

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)
```

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

Example

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

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

deleteCred

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

Description

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

Syntax

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

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

Example

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

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

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")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is deleted.
name	Specifies the name of the entitlement to delete.

Example

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

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

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")
```

Argument	Definition
appStripe	Specifies the application stripe where the resource is deleted.
name	Specifies the name of the resource deleted.
type	Specifies the type of resource deleted. The passed resource type *must* be present in the application stripe at the time this script is invoked.

Example

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

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

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)
```

Argument	Definition
appStripe	Specifies the application stripe from where to remove the resource type.
resourceTypeName	Specifies the name of the resource type to remove.

Example

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

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

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`.

Syntax

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

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run. **Note:** You can specify a `jps-config.xml` file or a `jps-config-jse.xml` file for the `jpsConfigFile` argument.
keyFilePath	Specifies the directory where the file `ewallet.p12` is created; note that the content of this file is encrypted and secured by the value passed to `keyFilePassword`.
keyFilePassword	Specifies the password to secure the file `ewallet.p12`; note that this same password must be used when importing that file.

Example

The following example writes the file `ewallet.p12` in the directory myDir:

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

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")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is located.
name	Specifies the name of the entitlement to access.

Example

The following example returns the information of the entitlement myEntitlement in the stripe myApplication:

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

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)
```

Argument	Definition
appStripe	Specifies the application stripe from where to fetch the resource type.
resourceTypeName	Specifies the name of the resource type to fetch.

Example

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

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

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)
```

Argument	Definition
appStripe	Specifies an application stripe.
appRoleName	Specifies a role name.
principalClass	Specifies the fully qualified name of a class.
principalName	Specifies the principal name. Set CN or DN attribute value for the user or groups from the LDAP server.

Example

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

- **Granting Application Role setting CN**

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

- **Granting Application Role setting DN**

```
wls:/mydomain/serverConfig> grantAppRole(appStripe="myApp",
 appRoleName="myRole",principalClass="com.example.xyzPrincipal",
principalName="cn=Admin,ou=IT,ou=Groups,dc=vm,dc=oracle,dc=com")
```

For more information, see Doc ID 2858916.1.

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")
```

Argument	Definition
appStripe	Specifies the application stripe where the principal resides.

Argument	Definition
principalClass	Specifies the class associated with the principal.
principalName	Specifies the name of the principal to which the entitlement is granted.
permSetName	Specifies the name of the entitlement granted.

Example

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

```
wls:/mydomain/serverConfig> grantEntitlement(appStripe="myApplication",
principalClass="oracle.security.jps.service.policystore.ApplicationRole",
principalName="myPrincipalName", permSetName="myEntitlement")
```

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])
```

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

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")
```

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
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run. **Note:** You can specify a `jps-config.xml` file or a `jps-config-jse.xml` file for the `jpsConfigFile` argument.
keyFilePath	Specifies the directory where the `ewallet.p12` is located.
keyFilePassword	Specifies the password used when the file `ewallet.p12` was generated.

Example

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

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
appStripe	Specifies an application stripe.

Example

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

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

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)
```

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

Example

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

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

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"])
```

Argument	Definition
configFile	Specifies the path to the OPSS configuration file. Optional. If specified, the script runs offline; the default context in the specified configuration file *must not* have a service instance reference to an identity store. If unspecified, the script runs online and it lists application stripes in the policy store.
regularExpression	Specifies the regular expression that returned stripe names should match. Optional. If unspecified, it matches all names. To match substrings, use the character *.

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:

```
wls:/mydomain/serverConfig> listAppStripes(configFile="
/home/myFile/jps-config.xml")
```

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

```
wls:/mydomain/serverConfig> listAppStripes(regularExpression="App*")
```

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"])
```

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

Example

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

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

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"])
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is deleted.
principalName	Specifies the name of the principal to match. Optional.
principalClass	Specifies the class of the principal to match. Optional.

Example

The following example lists all entitlements in the stripe myApplication:

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

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"])
```

Argument	Definition
appStripe	Specifies the application stripe from where to list entitlements.
resourceTypeName	Specifies the name of the type of the resources to list. Optional.
resourceName	Specifies the name of resource to match. Optional.

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")
```

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)
```

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

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:

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

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")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement resides.
permSetName	Specifies the name of the entitlement whose resources and actions to list.

Example

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

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

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"])
```

Argument	Definition
appStripe	Specifies the application stripe where the resources are listed.
type	Specifies the type of resource listed. The passed resource type *must* be present in the application stripe at the time this script is invoked.

Example

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

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

listResourceTypes

Online command that lists resource types.

Description

Lists all the resource types in a specified application stripe.

Syntax

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

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

Example

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

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

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")
```

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

Example

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

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

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

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

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. See Migrating with the Script migrateSecurityStore in *Securing Applications with Oracle Platform Security Services*.

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)
```

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run. **Note:** You can specify a `jps-config.xml` file or a `jps-config-jse.xml` file for the `jpsConfigFile` argument.
username	Specifies the distinguished name of the user in the LDAP store.
password	Specifies the password of the user.

Example

Let's assume that the password of the user with the distinguished name `cn=orcladmin` in the LDAP store has been changed to `password`, 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 `password`:

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

Any output regarding the audit service can be disregarded.

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*.

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)
```

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml` relative to the location where the command is run. **Note:** You can specify a `jps-config.xml` file or a `jps-config-jse.xml` file for the `jpsConfigFile` argument.

Example

```
restoreEncryptionKey(jpsConfigFile="pathName")
```

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)
```

Argument	Definition
appStripe	Specifies an application stripe.
appRoleName	Specifies a role name.
principalClass	Specifies the fully qualified name of a class.
principalName	Specifies the principal name. Set CN or DN attribute value for the user or groups from the LDAP server.

Example

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

- **Revoking Application Role setting CN**

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

- **Revoking Application Role setting DN**

```
wls:/mydomain/serverConfig> revokeAppRole(appStripe="myApp",
 appRoleName="myRole",principalClass="com.example.xyzPrincipal",
principalName="cn=Admin,ou=IT,ou=Groups,dc=vm,dc=oracle,dc=com")
```

For more information, see Doc ID 2858916.1.

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")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is deleted.
principalClass	Specifies the class associated with the principal.
principalName	Specifies the name of the principal to which the entitlement is revoked.
permSetName	Specifies the name of the entitlement deleted.

Example

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

```
wls:/mydomain/serverConfig> revokeEntitlement(appStripe="myApplication",
principalClass="oracle.security.jps.service.policystore.ApplicationRole",
principalName="myPrincipalName", permSetName="myEntitlement")
```

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])
```

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

Example

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

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

The following 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")
```

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")
```

Argument	Definition
appStripe	Specifies the application stripe where the entitlement is located.
name	Specifies the name of the entitlement to modify.
resourceName	Specifies the name of the resource to remove.
resourceType	Specifies the type of the resource to remove.
actions	Specifies the comma-separated list of actions to remove.

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")
```

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")
```

Argument	Definition
jpsConfigFile	Specifies the location of the file `jps-config.xml`; either relative to the location where the script is run, or the full path. **Note:** You can specify a `jps-config.xml` file or a `jps-config-jse.xml` file for the `jpsConfigFile` argument.

Example

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

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

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])
```

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

Example

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

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

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:

```
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 in *Securing Applications with Oracle Platform Security Services*.

Example

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

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

3
Audit Configuration WLST Commands

This chapter describes the Audit Configuration commands.
Use the WLST commands listed in Table 3-1 to view and manage audit policies and the audit repository configuration.

Table 3-1 WLST Audit Commands

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

See Introduction to Oracle Fusion Middleware Audit Framework in *Securing Applications with Oracle Platform Security Services*.

- createIAUView
- createAuditDBView
- deregisterAudit
- exportAuditConfig
- getIAUViewInfo
- getNonJavaEEAuditMBeanName
- getAuditPolicy
- getAuditRepository

- importAuditConfig
- listAuditComponents
- listAuditEvents
- setAuditPolicy
- setAuditRepository
- registerAudit

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])
```

Argument	Definition
componentType	The component whose definitions are the basis of the view.
viewType	The type of view; valid values are SIMPLE or INDEXABLE. Default is SIMPLE.

Examples

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

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

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

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])
```

Argument	Definition
fileName	The path and file name to which the SQL script is written.
componentType	The name of the registered component.
dbType	The database type. One of the following: DB_ORACLE, MS_SQL_SERVER, IBM_DB2.
viewType	The view type. One of the following: SIMPLE, INDEXABLE.

Example

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

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)
```

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

Example

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

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])
```

Argument	Definition
mbeanName	Specifies the name of the non-Java EE component MBean.
fileName	Specifies the path and file name to which the audit configuration should be exported.
componentType	Specifies that only events of the given component be exported to the file. If not specified, the audit configuration in `jps-config.xml` is exported.

Example

The following example exports the audit configuration for a component:

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

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

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

getIAUViewInfo

Returns information about the view of a component.

Description

Retrieves information about the view of a specified component.

Syntax

```
getIAUViewInfo(componentType)
```

Argument	Definition
componentType	The component whose definitions are the basis of the view.

Example

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

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

```
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:

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

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

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

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for non-Java EE components.
componentType	Requests the audit policy for a specific component registered in the audit store. If not specified, the audit policy in `jps-config.xml` is returned.

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(on='oracle.security.audit.test:type=CSAuditMBean,
name=CSAuditProxyMBean')
```

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
```

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])
```

Argument	Definition
mbeanName	Specifies the name of the non-Java EE component MBean.
fileName	Specifies the path and file name from which the audit configuration should be imported.
componentType	Specifies that only events of the given component be imported from the file. If not specified, the audit configuration in `jps-config.xml` is imported.

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')
```

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)
```

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

Example

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

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])
```

Argument	Definition
mbeanName	Specifies the name of the component MBean.
componentType	Specifies the component type to limit the list to all events of the component type.

Examples

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

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

Common Attributes
ComponentType
Type of the component. For MAS integrated SystemComponents this is the
componentType
InstanceId
```

```
Name of the MAS Instance, that this component belongs to
HostId
DNS hostname of originating host
HostNwaddr
IP or other network address of originating host
ModuleId
ID of the module that originated the message. Interpretation is unique within
Component ID.
ProcessId
ID of the process that originated the message
```

The following 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
HomeInstance
ECID
RID
...
```

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])
```

Argument	Definition
mbeanName	Specifies the name of the component audit MBean for non-Java EE components.
filterPreset	Specifies the filter preset to be changed.
addSpecialUsers	Specifies the special users to be added.
removeSpecialUsers	Specifies the special users to be removed.
addCustomEvents	Specifies the custom events to be added.
removeCustomEvents	Specifies the custom events to be removed.
componentType	Specifies the component definition type to be updated. The audit runtime policy for the component is registered in the audit store. If not specified, the audit configuration defined in jps-config.xml is modified.
maxFileSize	Specifies the maximum size of the log file.
andCriteria	Specifies the `and` criteria in a custom filter preset definition.
orCriteria	Specifies the `or` criteria in a custom filter preset definition.
componentEventsFile	Specifies a component definition file under the 11*g* Release 1 (11.1.1.6) metadata model. This parameter is required if you wish to create/update an audit policy in the audit store for an 11*g* Release 1 (11.1.1.6) metadata model component, and the filter preset level is set to “Custom".

Examples

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

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

wls:/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

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

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

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])
```

Argument	Definition
switchToDB	If `true`, switches the repository from file to database. Valid value: `true`.
dataSourceName	Specifies the JNDI name of the data source. This data source must be configured in the specified Oracle Weblogic Server domain.
interval	Specifies the time, in seconds, that the audit loader sleeps.
timezone	Specifies the time zone in which the audit loader records the timestamps of the audit events. Valid values are `utc` and `local`.

Argument	Definition
repostoryType	Specifies the database type to which the data has to be uploaded. The supported databases are Oracle, MS SQL Server and IBM DB2.
logDirectory	Specifies the audit log directory for SE applications to store bus stop files.
jdbcString	Specifies the audit repository jdbc connection string for SE applications.
dbUser	Specifies the audit repository IAU schema user.
interval	Specifies the audit repository IAU schema password.

Example

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:

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

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])
```

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

Example

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

4
OPSS Keystore Service Commands

This chapter describes the WLST 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. See Managing Keys and Certificates in *Securing Applications with Oracle Platform Security Services*.

Table 4-1 lists the WLST commands used to manage the keystore service.

Table 4-1 OPSS Keystore Service Commands

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

changeKeyPassword

Changes a key password.

Description

Changes the password for a key.

Syntax

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

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe containing the keystore
name	Specifies the name of the keystore
password	Specifies the keystore password
alias	Specifies the alias of the key entry whose password is changed
currentkeypassword	Specifies the current key password

Argument	Definition
newkeypassword	Specifies the new key password

Example

The following example changes the password on the key entry `orakey`:

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

changeKeyStorePassword

Changes the password of a keystore.

Description

Changes the password of the specified keystore.

Syntax

```
svc.changeKeyStorePassword(appStripe='stripe', name='keystore',
currentpassword='currentpassword', newpassword='newpassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe containing the keystore
name	Specifies the name of the keystore
currentpassword	Specifies the current keystore password
newpassword	Specifies the new keystore password

Example

The following example changes the password for `keystore2`.

```
wls:/mydomain/serverConfig> svc.changeKeyStorePassword(appStripe='system',
name='keystore2',
currentpassword='currentpassword', newpassword='newpassword')
```

createKeyStore

This keystore service command creates a new keystore.

Description

Creates a new keystore on the given application stripe.

Syntax

```
svc.createKeyStore(appStripe='stripe', name='keystore',
password='password',permission=true|false)
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore is created.
name	Specifies the name of the new keystore.
password	Specifies the keystore password.
permission	This parameter is true if the keystore is protected by permission only, false if protected by both permission and password.

Example

The following example creates a keystore named `keystore1`.

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

deleteKeyStore

Deletes the named keystore.

Description

This keystore service command deletes a specified keystore.

Syntax

```
svc.deleteKeyStore(appStripe='stripe', name='keystore', password='password')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore to be deleted.
password	Specifies the keystore password.

Example

The following example deletes the keystore named `keystore1`.

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

deleteKeyStoreEntry

Deletes a keystore entry.

Description

This command deletes the specified entry in a keystore.

Syntax

```
svc.deleteKeyStoreEntry(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the entry to be deleted
keypassword	Specifies the key password of the entry to be deleted

Example

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

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

exportKeyStore

Exports a keystore to a file.

Description

Exports a keystore to a specified file.

Syntax

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

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

Examples

The following example exports two aliases from the specified keystore.

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

exportKeyStoreCertificate

Exports a certificate.

Description

Exports a certificate, trusted certificate or certificate chain.

Syntax

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

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the entry to be exported
keypassword	Specifies the key password.
type	Specifies the type of keystore entry to be exported. Valid values are 'Certificate', 'TrustedCertificate' or 'CertificateChain'.
filepath	Specifies the absolute path of the file where certificate, trusted certificate or certificate chain is exported.

Example

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

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

exportKeyStoreCertificateRequest

Exports a certificate request.

Description

Generates and exports a certificate request from a keystore.

Syntax

```
svc.exportKeyStoreCertificateRequest(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword',
filepath='absolute_file_path')
```

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

Example

The following example exports a certificate request corresponding to the `orakey` alias.

```
wls:/mydomain/serverConfig>
svc.exportKeyStoreCertificateRequest(appStripe='system', name='keystore2',
password='password', alias='orakey', keypassword='keypassword',
filepath='/tmp/certreq.txt')
```

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

```
svc.generateKeyPair(appStripe='stripe', name='keystore', password='password',
dn='distinguishedname', keysize='keysize', alias='alias',
keypassword='keypassword'[, algorithm='algorithm'][,ext_san='ext_san'])
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
ext_san	Specifies the Subject Alternative Name (SAN) extension. The format for the argument is "type:value,...,type:value". Only the DNS type is supported.
dn	Specifies the distinguished name of the certificate wrapping the key pair.
keysize	Specifies the key size.
alias	Specifies the alias of the key pair entry.
keypassword	Specifies the key password.
algorithm	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 uses the RSA algorithm.

Examples

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

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

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

```
svc.generateKeyPair(appStripe='system', name='keystore2',
password='<password>', dn='cn=www.oracle.com', keysize='2048',
alias='orakey', keypassword='<keypassword>',
ext_san='DNS:server1.oracle.com,DNS:www.oracle.com')
```

generateSecretKey

Generates a secret key. This command creates only a symmetric key, not a public/private key pair. To view the properties after creating the symmetric key, use getKeyStoreSecretKeyProperties.

Description

Generates a symmetric key in a keystore.

Syntax

```
svc.generateSecretKey(appStripe='stripe', name='keystore', password='password',
algorithm='algorithm', keysize='keysize', alias='alias',
keypassword='keypassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
algorithm	Specifies the symmetric key algorithm.
keysize	Specifies the key size.
alias	Specifies the alias of the key entry.
keypassword	Specifies the key password.

Example

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

```
wls:/mydomain/serverConfig> svc.generateSecretKey(appStripe='system',
name='keystore2', password='password',
algorithm='AES', keysize='128', alias='seckey', keypassword='keypassword')
```

getKeyStoreCertificates

Gets a certificate from the keystore. Use this command to view the contents of the public key and X509 certificate that you have imported from a keystore (supported keystore types are JKS or JCEKS), or that you have created using the generateKeyPair command.

Description

Retrieves information about a certificate or trusted certificate.

Syntax

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

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the certificate, trusted certificate or certificate chain to be displayed.
keypassword	Specifies the key password.

Example

The following example gets certificates associated with `keystore3`.

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

getKeyStoreSecretKeyProperties

Retrieves secret key properties.

Description

Retrieves secret key properties like the algorithm.

Syntax

```
svc.getKeyStoreSecretKeyProperties(appStripe='stripe', name='keystore',
password='password', alias='alias', keypassword='keypassword')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the secret key whose properties are displayed.
keypassword	Specifies the secret key password.

Example

The following example gets properties for secret key `seckey`:

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

importKeyStore

Imports a keystore from file. This command imports any public key, private key, symmetric key, and trusted certificates from the key store file into OPSS Keystore Service keystore.

Description

Imports a keystore from a system file.

Syntax

```
svc.importKeyStore(appStripe='stripe', name='keystore', password='password',
aliases='comma-separated-aliases', keypasswords='comma-separated-keypasswords',
type='keystore-type', permission=true|false, filepath='absolute_file_path')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore will reside.
name	Specifies the name of the keystore.

Argument	Definition
password	Specifies the keystore password. These rules apply: • If importing an auto-login Oracle Wallet file, no password is needed. • If importing a password-protected Oracle Wallet file (ewallet.p12), enter a password of minimum eight characters.
aliases	Specifies the comma-separated aliases of the entries to be imported from the file. If type is set to OracleWallet, it is not required; otherwise, it is a required argument.
keypasswords	Specifies the passwords of the keys in the file. These rules apply: • If type is JKS or JCEKS, enter comma-separated passwords of the keys. • If `type` is OracleWallet, no password is needed. The key passwords will be the same as the keystore password.
type	Specifies the imported keystore type. Valid values are 'JKS' or 'JCEKS' or 'OracleWallet'.
filepath	If `type` is set to JKS or JCEKS, it specifies rthe absolute path of the keystore file to be imported, including filname. If type is set to OracleWallet, it specifies the absolute path of the directory where the Oracle Wallet is located.
permission	Specifies true if keystore is protected by permission only, false if protected by both permission and password. If set to true, the imported file is permission protected, so when call getKeyStore or getKey, set password to null.

Example

The following example imports a JKS keystore file to `keystore2`:

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

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

importKeyStoreCertificate

Imports a certificate or other specified object.

Description

Imports a certificate, trusted certificate or certificate chain.

Syntax

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

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
alias	Specifies the alias of the entry to be imported.
keypassword	Specifies the key password of the newly imported entry.
type	Specifies the type of keystore entry to be imported. Valid values are 'Certificate', 'TrustedCertificate' or 'CertificateChain'.
filepath	Specifies the absolute path of the file from where certificate, trusted certificate or certificate chain is imported.

Example

The following example imports a certificate into `keystore2`.

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

listExpiringCertificates

Lists expiring certificates.

Description

Lists expiring certificates and optionally renews them.

Syntax

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

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

Argument	Definition
days	Specifies that the list should only include certificates within this many days from expiration.
autorenew	Specifies true for automatically renewing expiring certificates, false for only listing them.

Example

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

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

listKeyStoreAliases

Lists the aliases in a keystore.

Description

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

Syntax

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

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe where the keystore resides.
name	Specifies the name of the keystore.
password	Specifies the keystore password.
type	Specifies the type of entry for which aliases are listed. Valid values are 'Certificate', 'TrustedCertificate', 'SecretKey' or '*'.

Example

The following example lists secret keys in `keystore2`:

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

listKeyStores

Lists all the keystores in a stripe.

Description

Lists all the keystores in the specified stripe.

Syntax

```
svc.listKeyStores(appStripe='stripe')
```

Argument	Definition
svc	Specifies the service command object obtained through a call to getOpssService().
appStripe	Specifies the name of the stripe whose keystores are listed.

Example

The following example lists all keystores on all stripes.

```
wls:/mydomain/serverConfig> svc.listKeyStores(appStripe='*')
```

syncKeyStores

Synchronizes keystores from the OPSS security store to the local repository.

Description

Downloads keystores from an application stripe in the security store to the specified directory on the file system, in the given format.

If the target format is Oracle Wallet, the command downloads the contents of all KSS keystores for a given stripe into auto-login wallets on the server. The contents of the domain trust store are automatically included in each wallet.

Syntax

The syntax is as follows:

```
syncKeyStores(appStripe='<application_stripe>',
keystoreFormat='exported_file_format',
rootDirectory='root_dir_absolute_path')
```

Argument	Definition
appStripe	Name of the KSS application stripe containing the keystores that need to be synchronized with the local repository.
keystoreFormat	Specifies the format of the target keystore. Valid formats are 'KSS' and 'OracleWallet'. If the `keystoreFormat` is 'OracleWallet', then the keystores in this stripe must be permission-protected only. You cannot use password-protected keystores in an Oracle wallet.
rootDirectory	For the Oracle Wallet format, specifies the absolute path of the server directory where the wallet(s) are created.

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

Example

The following example looks up the security store for the "system" stripe and downloads its contents into the `keystores.xml` file under the *DOMAIN_HOME*`/config/fmwconfig` directory.

```
wls:/mydomain/serverConfig> syncKeyStores((appStripe='system', keystoreFormat='KSS')
```

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

```
syncKeyStores(appStripe="ohs",
keystoreFormat="OracleWallet", rootDirectory="/tmp/bin")
```

5

SSL Configuration WLST Commands

This chapter describes SSL configuration WLST commands.
This chapter contains the following sections:

- About SSL Configuration Commands
- Properties Files for SSL

About SSL Configuration Commands

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

Use the commands listed in Table 5-1 for this task.

> **See Also:**
>
> Command-Line Interface for Keystores and Wallets in *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 5-1 WLST Commands for SSL Configuration

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

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`.

- Structure of Properties Files
- Examples of Properties Files

Structure of Properties Files

All the SSL properties files have a consistent structure.

Table 5-2 provides details about the key-value structure and usage of these files.

Table 5-2 Parameters in Properties File

Key	Mandatory?	Allowed Values for Oracle HTTP Server	Usage
SSLEnabled	No	true false	Either value

Table 5-2 (Cont.) Parameters in Properties File

Key	Mandatory?	Allowed Values for Oracle HTTP Server	Usage
Ciphers	No	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	One or more comma separated values
SSLVersions	No	nzos_Version_3_0 nzos_Version_3_0_With_2_0_Hello nzos_Version_1_0	One or more comma separated values
CertValidation	No	none crl	Either value
CertValidation Path	No	file://`crl_file_path` dir://`crl_dir_path`	Path of the CRL file, or directory containing CRL files
KeyStore	No	Valid wallet name	
TrustStore	No	N/A	
AuthenticationType	No	None Server Optional Mutual	Any one value

Table 5-3 shows the default values:

Table 5-3 Default Values of Parameters

Key	Default Value for Oracle HTTP Server
SSLEnabled	true
Ciphers	null
SSLVersions	null
CertValidation	none
CertValidation Path	null
KeyStore	default
TrustStore	-

Table 5-3 (Cont.) Default Values of Parameters

Key	Default Value for Oracle HTTP Server
Authentication Type	Server

Note:

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

Examples of Properties Files

Some examples demonstrating the use of the properties files follow.

Example 1: Basic Properties File

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

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

Example 2: Basic Properties File

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

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

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

- The setting:

  ```
  SSLEnabled=true
  ```

  Here the value `true` is explicitly specified.

- The setting:

  ```
  SSLEnabled=
  ```

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

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

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

Example 3: Properties File with Version for Oracle HTTP Server

```
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 Properties Files for SSL.

Syntax

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

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ohs'.
listener	Specifies the name of the component listener to be configured for SSL.
filePath	Specifies the absolute path of the properties file containing the SSL attributes to set.

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')
```

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

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')
```

6

Wallet Configuration WLST Commands

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

The WLST Wallet Commands

WLST commands allow to manage Oracle wallets for Oracle Fusion Middleware components. Table 6-1 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 6-1 WLST Commands for Oracle Wallet Management

Use this command...	To...	Use with WLST...
addCertificateRequest	Generate a certificate signing request in an Oracle wallet.	Online
addSelfSignedCertificate	Add a self-signed certificate to an Oracle wallet.	Online
changeWalletPassword	Change the password to an Oracle wallet.	Online
createWallet	Create an Oracle wallet.	Online
deleteWallet	Delete an Oracle wallet.	Online
exportWallet	Export an Oracle wallet to a file.	Online
exportWalletObject	Export an object (for example, a certificate) from an Oracle wallet to a file.	Online
getWalletObject	Display a certificate or other object present in an Oracle wallet.	Online
importWallet	Import an Oracle wallet from a file.	Online
importWalletObject	Import a certificate or other object from a file to an Oracle wallet.	Online
listWalletObjects	List all objects (such as certificates) present in an Oracle wallet.	Online
listWallets	List all Oracle wallets configured for a component instance.	Online
removeWalletObject	Remove a certificate or other object from a component instance's Oracle wallet.	Online

See Also:

Command-Line Interface for Keystores and Wallets in *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')
```

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

Example

The following command generates a certificate signing request with DN `cn=www.example.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.example.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')
```

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

Example

The following command adds a self-signed certificate with DN `cn=www.example.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.example.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')
```

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

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')
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ohs'.
walletName	Specifies the name of the wallet file to be created.
password	Specifies the wallet password.

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')
```

Argument	Definition
instName	Specifies the name of the application server instance.
compName	Specifies the name of the component instance.
compType	Specifies the type of component. Valid value is 'ohs'.
walletName	Specifies the name of the wallet file to be deleted.

Example

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

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

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')
```

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

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')
```

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

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

Example

The following command exports a certificate signing request with DN `cn=www.example.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.example.com')
```

The following command exports a certificate with DN `cn=www.example.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.example.com')
```

The following command exports a trusted certificate with DN `cn=www.example.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.example.com')
```

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

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

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

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`:

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

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

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

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

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

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

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

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

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:

```
wls:/mydomain/serverConfig> importWallet('inst1', 'ohs1', 'ohs', 'wallet1', '', '/tmp/
cwallet.sso')
```

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

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

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')
```

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

Argument	Definition
type	Specifies the type of wallet object to be imported. Valid values are 'Certificate', 'TrustedCertificate' and 'TrustedChain'.
filePath	Specifies the absolute path of the file containing the wallet object.

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')
```

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

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')
```

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

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')
```

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

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.example.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.example.com')
```

The following command removes a certificate indicated by DN `cn=www.example.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.example.com')
```

The following command removes a trusted certificate indicated by DN `cn=www.example.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.example.com')
```