Oracle® Fusion Middleware

WebLogic Scripting Tool Command Reference for Identity and Access Management

11*g* Release 2 (11.1.2.3.0)

E57375-04

January 2017

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

ORACLE

Oracle Fusion Middleware WebLogic Scripting Tool Command Reference for Identity and Access Management, 11*g* Release 2 (11.1.2.3.0)

E57375-04

Primary Author: Binitha Monnappa

Contents

Preface

This preface describes the document accessibility features and conventions used in this guide—*WebLogic Scripting Tool Command Reference for Identity and Access Management*.

Audience

This document is intended for Administrators who are familiar with:

- Access and Identity Management concepts and administration
- Oracle WebLogic Server concepts and administration
- LDAP server concepts and administration
- Database concepts and administration (for policy and session management data)
- Web server concepts and administration
- WebGate and mod_osso agents
- Auditing, logging, and monitoring concepts
- Security token concepts
- Integration of the Policy store, Identity store, and familiarity with OIS might be required

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Fusion Middleware 11*g* Release 2 (11.1.2.3.0) documentation set:

- *Oracle Access Management 11g* Release 2 (11.1.2.3.0) *Release Notes*

- *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*—Explains how to use the Oracle Universal Installer and the WebLogic Configuration Wizard for initial Oracle Access Management 11g deployment. Installing 11g WebGates for Access Manager is also covered.
- *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*—Explains how to manage configuration and policies for Access Manager, Security Token Service, Identity Federation, Mobile and Social, Identity Context and other Oracle Access Management suite services.
- *Oracle Fusion Middleware Developer's Guide for Oracle Access Management*—Explains how to write custom AccessGates and plug-ins that enable programmatic access to extend Access Manager single sign-on and authorization functions.
- *Oracle Fusion Middleware Upgrade Guide for Java EE*—For information about the types of Java EE environments available in 10g and instructions for upgrading those environments to Oracle Fusion Middleware 11g.
- *Oracle Fusion Middleware Upgrade Guide for Oracle Identity and Access Management*—Explains how to upgrade Oracle Identity Management 10g components to Oracle Identity Management 11g.
- *Oracle Fusion Middleware Performance and Tuning Guide*—Explains how to monitor and optimize performance, configure components for optimal performance, and write highly performant applications in the Oracle Fusion Middleware environment.
- *Oracle Fusion Middleware Administrator's Guide*—Describes how to manage a secure Oracle Fusion Middleware environment, including how to change ports, deploy applications, and how to back up and recover Oracle Fusion Middleware. This guide also explains how to move data from a test to a production environment.
- *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Identity Management*—For a step-by-step guide to deployment.
- *Oracle Fusion Middleware High Availability Guide*—For high availability conceptual information as well as administration and configuration procedures for Administrators, developers, and others whose role is to deploy and manage Oracle Fusion Middleware with high availability requirements.
- *Security and Administrator's Guide for Web Services*—Describes how to administer and secure Web services.

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

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

- Guide to This Document
- Document Scope and Audience
- Using WSLT Commands
- Related Documentation

1.1 Guide to This Document

This document is organized as follows:

- This chapter introduces the guide and lists related documentation.
- Part I, "WLST for Oracle Identity and Access Management" summarizes WebLogic Scripting Tool (WLST) commands and variables for Identity and Access Management.
- Part II, "WLST for Oracle Mobile Security Suite" provides detailed descriptions for commands and variables for the Oracle Mobile Security Suite gateway component.

A refreshed version of this 11.1.2.3.0 documentation (published in September 2015) was reorganized. For example, commands previously collected in one long chapter in Part I, "WLST for Oracle Identity and Access Management" are now grouped in several shorter chapters.

1.2 Document Scope and Audience

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

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

This document is written for WebLogic Server administrators and operators who deploy Java EE applications using the Java Platform, Enterprise Edition (Java EE) from

Oracle. It is assumed that readers are familiar with Web technologies and the operating system and platform on which WebLogic Server is installed.

1.3 Using WSLT Commands

To use the custom WLST commands on WebLogic Server, you must invoke the WLST script from the Oracle Common home. See "Using Custom WLST Commands" in the *Oracle Fusion Middleware Administrator's Guide.* To use the applicable Infrastructure Security custom WLST commands on a WebSphere Server, see the 3rd Party Integration Guide. For other WebLogic Server and Fusion Middleware WLST commands, refer to the WLST Command Reference guide for your installed release of WebLogic Server.

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

1.4 Related Documentation

WLST is one of several interfaces for managing and monitoring WebLogic Server. For information about how to use the WebLogic Scripting Tool, refer to *Oracle WebLogic Scripting Tool*. For information about the other management interfaces, see:

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

Part I

WLST for Oracle Identity and Access Management

Part I describes the Identity and Access Management WLST commands.

2

Oracle Fusion Middleware SSL WLST Commands

This chapter provides descriptions of custom WebLogic Scripting Tool (WLST) commands for Oracle Fusion Middleware SSL, including command syntax, arguments and examples.

The following section lists the Oracle Fusion Middleware SSL WLST commands and contains links to the command reference details.

- SSL Configuration Commands

2.1 SSL Configuration Commands

Use the WLST commands listed in Table 2–1 to view and manage SSL configuration for Oracle Fusion Middleware components.

Table 2–1 WLST Commands for SSL Configuration

Use this command...	To...	Use with WLST...
addCertificateRequest	Generate a certificate signing request in an Oracle wallet.	Online
addSelfSignedCertificate	Add a self-signed certificate to an Oracle wallet.	Online
changeKeyStorePassword	Change the password to a JKS keystore.	Online
changeWalletPassword	Change the password to an Oracle wallet.	Online
configureSSL	Set the SSL attributes for a component listener.	Online
createKeyStore	Create a JKS keystore.	Online
createWallet	Create an Oracle wallet.	Online
deleteKeyStore	Delete a JKS keystore.	Online
deleteWallet	Delete an Oracle wallet.	Online
exportKeyStore	Export a JKS keystore to a file.	Online
exportKeyStoreObject	Export an object from a JKS keystore to a file.	Online
exportWallet	Export an Oracle wallet to a file.	Online
exportWalletObject	Export an object from an Oracle wallet to a file.	Online
generateKey	Generate a key pair in a JKS keystore.	Online
getKeyStoreObject	Display a certificate or other object present in a JKS keystore.	Online

Table 2–1 (Cont.) WLST Commands for SSL Configuration

Use this command...	To...	Use with WLST...
getSSL	Display the SSL attributes for a component listener.	Online
getWalletObject	Display a certificate or other object present in an Oracle wallet.	Online
importKeyStore	Import a JKS keystore from a file.	Online
importKeyStoreObject	Import a certificate or other object from a file to a JKS keystore.	Online
importWallet	Import an Oracle wallet from a file.	Online
importWalletObject	Import a certificate or other object from a file to an Oracle wallet.	Online
listKeyStoreObjects	List all objects present in a JKS keystore.	Online
listKeyStores	List all JKS keystores configured for a component instance.	Online
listWalletObjects	List all objects present in an Oracle wallet.	Online
listWallets	List all Oracle wallets configured for a component instance.	Online
removeKeyStoreObject	Remove a certificate or other object from a component instance's JKS keystore.	Online
removeWalletObject	Remove a certificate or other object from a component instance's Oracle wallet.	Online

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

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, Oracle WebCache or Oracle Internet Directory). 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 values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
DN	Specifies the Distinguished Name of the key pair entry.
keySize	Specifies the key size in bits.

Example

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

```
wls:/mydomain/serverConfig> addCertificateRequest('inst1', 'oid1',
'oid','wallet1', 'password', 'cn=www.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, Oracle WebCache or Oracle Internet Directory). 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 values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
DN	Specifies the Distinguished Name of the key pair entry.
keySize	Specifies the key size in bits.

Example

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

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

changeKeyStorePassword

Online command that changes the keystore password.

Description

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

Syntax

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

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

Example

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

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

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, Oracle WebCache or Oracle Internet Directory). 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 values are 'oid', 'ohs', and 'webcache'.
walletName	Specifies the file name 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')
```

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, default attribute values are used. For component-specific SSL attribute value defaults, see the chapter "SSL Configuration in Oracle Fusion Middleware" in the *Oracle Fusion Middleware Administrator's Guide*.

Syntax

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

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

Examples

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

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

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

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

createKeyStore

Online command that creates a JKS keystore.

Description

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

Syntax

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

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

Example

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

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

createWallet

Online command that creates an Oracle wallet.

Description

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

Syntax

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

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

Examples

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

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

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

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

deleteKeyStore

Online command that deletes a keystore.

Description

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

Syntax

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

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

Example

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

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

deleteWallet

Online command that deletes an Oracle wallet.

Description

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

Syntax

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

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

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')
```

exportKeyStore

Online command that exports the keystore to a file.

Description

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

Syntax

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

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

Example

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

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

exportKeyStoreObject

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

Description

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

Syntax

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

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

Examples

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

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

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

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

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

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

exportWallet

Online command that exports an Oracle wallet.

Description

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

Syntax

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

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

Examples

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

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

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

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

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 (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). DN is used to indicate 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.
compType	Specifies the type of component. Valid values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
type	Specifies the type of wallet object to be exported. Valid values are 'CertificateRequest', 'Certificate', 'TrustedCertificate' or 'TrustedChain'.
path	Specifies the absolute path of the directory under which the object is exported as a file base64.txt.
DN	Specifies the Distinguished Name of the wallet object being exported.

Examples

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

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

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

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

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

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

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

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

generateKey

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

Description

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

Syntax

```
generateKey(instName, compName, compType, keystoreName, password, DN, keySize,
alias, algorithm)
```

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

Examples

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

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

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

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

getKeyStoreObject

Online command that shows details about a keystore object.

Description

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

Syntax

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

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

Examples

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

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

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

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

getSSL

Online command that lists the configured SSL attributes.

Description

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

Syntax

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

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

Example

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

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

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 (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). The wallet object is indicated by its index number, as given by the `listWalletObjects` command. For certificates or trusted certificates, it shows the certificate details including DN, key size, algorithm and other data. For certificate signing requests, it shows the subject DN, key size and algorithm.

Syntax

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

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

Examples

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

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

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

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

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

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

importKeyStore

Online command that imports a keystore from a file.

Description

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

Syntax

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

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

Example

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

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

importKeyStoreObject

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

Description

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

Syntax

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

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

Examples

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

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

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

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

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

Syntax

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

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

Examples

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

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

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

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

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 (Oracle HTTP Server, Oracle WebCache component or Oracle Internet Directory).When importing a certificate, use the same wallet file from which the certificate signing request was generated.

Syntax

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

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

Examples

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

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

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

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

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

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

listKeyStoreObjects

Online command that lists the contents of a keystore.

Description

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

Syntax

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

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

Examples

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

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

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

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

listKeyStores

Online command that lists all the keystores for a component.

Description

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

Syntax

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

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

Example

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

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

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 (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory).

Syntax

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

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

Examples

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

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

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

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

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

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

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 (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory), 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 values are 'ohs', 'oid', and 'webcache'.

Example

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

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

removeKeyStoreObject

Online command that removes an object from a keystore.

Description

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

Syntax

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

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

Examples

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

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

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

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

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

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

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 (Oracle HTTP Server, Oracle WebCache or Oracle Internet Directory). 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 values are 'ohs', 'oid', and 'webcache'.
walletName	Specifies the name of the wallet file.
password	Specifies the password of the wallet.
type	Specifies the type of the keystore object to be removed. Valid values are 'CertificateRequest', 'Certificate', 'TrustedCertificate' or 'TrustedAll'.
DN	Specifies the Distinguished Name of the wallet object to be removed.

Examples

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

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

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

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

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

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

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

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

3

Policy and Credential WLST Commands

This chapter provides descriptions of custom WebLogic Scripting Tool (WLST) commands to administer policies and credentials for Oracle Access Management Access Manager, including command syntax, arguments and examples.

The following section lists the policy and credential WLST commands and contains links to the command reference details.

- Policy and Credential Commands

> **Note:** The name of this section (now chapter) has been changed from previous versions of this book to more clearly reflect its contents. The commands and information remain the same.

3.1 Policy and Credential Commands

Use the WLST commands listed in Table 3–1 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 3–1 WLST Security Commands

Use this command...	To...	Use with WLST...
listAppStripes	List application stripes in policy store.	Online
createAppRole	Create a new application role.	Online
deleteAppRole	Remove an application role.	Online
grantAppRole	Add a principal to a role.	Online
revokeAppRole	Remove a principal from a role.	Online
listAppRoles	List all roles in an application.	Online
listAppRolesMembers	List all members in an application role.	Online
grantPermission	Create a new permission.	Online
revokePermission	Remove a permission.	Online
listPermissions	List all permissions granted to a principal.	Online
deleteAppPolicies	Remove all policies in an application.	Online
migrateSecurityStore	Migrate policies or credentials from a source repository to a target repository.	Offline

Table 3–1 (Cont.) WLST Security Commands

Use this command...	To...	Use with WLST...
listCred (Deprecated)	Obtain the list of attribute values of a credential.	Online
updateCred	Modify the attribute values of a credential.	Online
createCred	Create a new credential.	Online
deleteCred	Remove a credential.	Online
modifyBootStrapCredential	Update bootstrap credential store.	Offline
addBootStrapCredential	Add a credential to the bootstrap credential store.	Offline
exportEncryptionKey	Export the domain encryption key to the file `ewallet.p12`.	Offline
importEncryptionKey	Import the encryption key in file `ewallet.p12` to the domain.	Offline
restoreEncryptionKey	Restore the domain encryption key as it was before the last importing.	Offline
reassociateSecurityStore	Reassociate policies and credentials to an LDAP repository.	Online
upgradeSecurityStore	Upgrade security data from data used with release 10.1.x to data used with release 11.	Offline
createResourceType	Create a new resource type.	Online
getResourceType	Fetch an existing resource type.	Online
deleteResourceType	Remove an existing resource type.	Online
createResource	Create a resource.	Online
deleteResource	Remove a resource.	Online
listResources	List resources in an application stripe.	Online
listResourceActions	List actions in a resource.	Online
createEntitlement	Create an entitlement.	Online
getEntitlement	List an entitlement.	Online
deleteEntitlement	Remove an entitlement.	Online
addResourceToEntitlement	Add a resource to an entitlement.	Online
revokeResourceFromEntitlement	Remove a resource from an entitlement	Online
listEntitlements	List entitlements in an application stripe.	Online
grantEntitlement	Create an entitlement.	Online
revokeEntitlement	Remove an entitlement.	Online
listEntitlement	List an entitlement.	Online
listResourceTypes	List resource types in an application stripe.	Online

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

Example

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

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

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

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

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.

Example

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

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

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.

Example

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

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

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

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

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

Optional arguments are enclosed in between square brackets.

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

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

Examples

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

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

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

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

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

Optional arguments are enclosed in between square brackets.

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

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

Examples

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

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

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

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

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

Optional arguments are enclosed in between square brackets.

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

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

Examples

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

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

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

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

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 invocation removes all policies of application `myApp`:

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

migrateSecurityStore

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

Description

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

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

Syntax

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

Optional arguments are enclosed in square brackets.

To migrate identities, use the following syntax:

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

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

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

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

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

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

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

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

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

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

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

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

Argument	Definition
mode	Specifies whether the migration should stop and signal an error upon encountering a duplicate principal or a duplicate permission in an application policy. Set to lax to allow the migration to continue upon encountering duplicate items, to migrate just one of the duplicated items, and to log a warning to this effect; set to strict to force the migration to stop upon encountering duplicate items. If unspecified, it defaults to strict.

Note the following requirements about the passed arguments:

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

Example

The following invocation illustrates the migration of the file-based policies of application `PolicyServlet1` to file-based policies of application `PolicyServlet2`, that does not stop on encountering duplicate principals or permissions, that migrates just one of duplicate items, and that logs a warning when duplicates are found:

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

The above invocation assumes that:

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

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

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

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

```
wls:/mydomain/serverConfig> migrateSecurityStore(type="credStore",
```

```
configFile="jps-congif.xml", src="default1", dst="context2")
```

The above invocation assumes that:

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

```
<serviceInstance name="credstore1" provider="some.provider">
  <property name="location" value="./credstore1/cwallet.sso"/>
</serviceInstance>
<serviceInstance name="credstore2" provider="some.provider">
  <property name="location" value="./credstore2/cwallet.sso"/>
</serviceInstance>
...
<jpsContext name="default1">
  <serviceInstanceRef ref="credstore1"/>
  ...
</jpsContext>
<jpsContext name="context2">
  <serviceInstanceRef ref="credstore2"/>
  ...
</jpsContext>
```

For detailed configuration examples to use with this command, see *Oracle Fusion Middleware Security Guide*.

listCred (Deprecated)

The `listCred` command has been deprecated. This functionality should be done using the Oracle Enterprise Manager Console. See the Oracle Enterprise Manager documentation for details. An example of how to retrieve the OAM Keystore and alias can be found in the *Oracle Fusion Middleware Administrator's Guide for Oracle Access Management*.

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

Optional arguments are enclosed in square brackets.

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

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

Example

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

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

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

Optional arguments are enclosed in square brackets.

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

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

Example

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

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

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

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

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.
username	Specifies the distinguished name of the user in the LDAP store.
password	Specifies the password of the user.

Example

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

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

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

Any output regarding the audit service can be disregarded.

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.
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 invocation adds a credential to the bootstrap credential store:

```
wls:/mydomain/serverConfig>
addBootStrapCredential(jpsConfigFile='./jps-config.xml', map='myMapName',
key='myKeyName', username='myUser', password='myPassword')
```

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.
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 invocation writes the file `ewallet.p12` in the directory myDir:

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

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.
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")
```

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.

Example

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

reassociateSecurityStore

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

Description

Migrates, within a give domain, *both* the policy store and the credential store to a target LDAP server repository. The only kinds of LDAP servers allowed are OID or OVD. This command also allows setting up a policy store shared by different domains (see optional argument `join` below). In the event of an error, the command returns a `WLSTException`. This command runs in interactive mode only.

Syntax

```
reassociateSecurityStore(domain, admin, password, ldapurl, servertype, jpsroot [,
join] [,keyFilePath, keyFilePassword])
```

Argument	Definition
domain	Specifies the domain name where the reassociating takes place.
admin	Specifies the administrator's user name on the LDAP server. The format is `cn=usrName`.
password	Specifies the password associated with the user specified for the argument `admin`.
ldapurl	Specifies the URI of the LDAP server. The format is `ldap//:host:port`, if you are using a default port, or `ldaps://host:port`, if you are using a secure LDAP port. The secure port must be configured specially for this function and it is distinct from the default (non-secure) port.
servertype	Specifies the kind of the target LDAP server. The only valid types are OID or OVD.
jpsroot	Specifies the root node in the target LDAP repository under which all data is migrated. The format is `cn=nodeName`.
join	Specifies whether the domain is to share a policy store specified in some other domain. Optional. Set to true to share an existing policy store in another domain; set to false otherwise. If unspecified, it defaults to false. The use of this argument allows multiple WebLogic domains to point to the same logical policy store.
keyFilePath	Specifies the directory where the `ewallet.p12` is located.
keyFilePassword	Specifies the password used when the file `ewallet.p12` was generated.

Examples

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

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

Suppose that you want some *other* domain (distinct from `myDomain`, say `otherDomain`) to share the policy store in `myDomain`. Then you would invoke the command as follows:

```
wls:/mydomain/serverConfig> reassociateSecurityStore(domain="otherDomain",
```

```
admin="cn=adminName", password="myPass", ldapurl="ldap://myhost.example.com:3060",
servertype="OID", jpsroot="cn=testNode", join="true")
```

upgradeSecurityStore

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

Description

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

Syntax

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

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

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

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

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

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

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

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

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

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

Argument	Definition
dst	Specifies the name of the jpsContext in the file passed to the argument jpsConfigFile where the destination store is configured. Optional. If unspecified, it defaults to the default context in the file passed in the argument jpsConfigFile.

Examples

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

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

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

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

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. Optional arguments are enclosed in between square brackets; all other arguments are required. In the event of an error, the command returns a `WLSTException`.

Syntax

Optional arguments are enclosed in square brackets.

```
createResourceType(appStripe, resourceTypeName, displayName, description [,
provider] [, matcher], actions [, delimeter])
```

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

Example

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

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

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 invocation fetches the resource type myResType from the stripe myApplication:

```
wls:/mydomain/serverConfig> getResourceType(appStripe="myApplication",
resourceTypeName="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 invocation removes the resource type myResType from the stripe myApplication:

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

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.

If this command is used in offline mode after reassociating to a DB-based store, the configuration file produced by the reassociation *must* be manually edited as described in "Running listAppStripes after Reassociating to a DB-Based Store" in *Oracle Fusion Middleware Security Guide*.

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 *.

Examples

The following (online) invocation returns the list of application stripes in the policy store:

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

The following (offline) invocation returns the list of application stripes in the policy store referenced in the default context of the specified configuration file:

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

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

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

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 invocation creates the resource myResource in the stripe myApplication:

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

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 invocation deletes the resource myResource in the stripe myApplication:

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

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 invocation lists all resources of type myResType in the stripe myApplication:

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

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 invocation lists the resources and actions of the entitlement myEntitlement in the stripe myApplication:

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

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.
resourceName	Specifies the name of the one resource member of the entitlement created.
actions	Specifies a comma-delimited 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 invocation creates the entitlement myEntitlement with just the resource myResource in the stripe myApplication:

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

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 invocation returns the information of the entitlement myEntitlement in the stripe myApplication:

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

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 invocation deletes the entitlement myEntitlement in the stripe myApplication:

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

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-delimited list of actions for the added resource.

Example

The following invocation adds the resource myResource to the entitlement myEntitlement in the application stripe myApplication:

```
wls:/mydomain/serverConfig> addResourceToEntitlement(appStripe="myApplication",
name="myEntitlement", resourceName="myResource", resourceType="myResType",
actions="view,edit")
```

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-delimited list of actions to remove.

Example

The following invocation removes the resource myResource from the entitlement myEntitlement in the stripe myApplication:

```
wls:/mydomain/serverConfig>
revokeResourceFromEntitlement(appStripe="myApplication", name="myEntitlement",
resourceName="myResource", resourceType="myResType", actions="view,edit")
```

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 invocation lists all the entitlements in the stripe myApplication:

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

The following invocation lists all the entitlements in the stripe myApplication that contain a resource type myResType and a resource whose name match the resource name myResName:

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

grantEntitlement

Online command that creates a new entitlement.

Description

Creates a new entitlement with 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 entitlement is created.
principalClass	Specifies the class associated with the principal.
principalName	Specifies the name of the principal to which the entitlement is granted.
permSetName	Specifies the name of the entitlement created.

Example

The following invocation creates the entitlement myEntitlement in the stripe myApplication:

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

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 invocation deleted the entitlement myEntitlement in the stripe myApplication:

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

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 invocation lists all entitlements in the stripe myApplication:

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

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 invocation lists all resource types in the stripe myApplication:

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

4

Access Manager WLST Commands

This chapter provides descriptions of custom WebLogic Scripting Tool (WLST) commands for Oracle Access Management Access Manager, including command syntax, arguments and examples.

The following section lists the Oracle Access Management Access Manager WLST commands and contains links to the command reference details.

- Access Manager Commands

4.1 Access Manager Commands

Use the WLST commands listed in Table 4–1 to manage Oracle Access Management Access Manager (Access Manager) related components, such as authorization providers, identity asserters, and SSO providers. Other commands allow you to display metrics and deployment topology, manage your server and agent configurations and logger settings.

Table 4–1 WLST Access Manager Commands

Use this command...	To...	Use with WLST...
displayAuthZCallBackKey	Generate and retrieve the key used to hash a resource URL in an authorization policy.	Online
updateCustomPages	Enables and disables custom error and login pages.	Online Offline
createUserIdentityStore	Create a user identity store registration.	Online Offline
editUserIdentityStore	Edit a user identity store registration.	Online Offline
deleteUserIdentityStore	Delete a user identity store registration.	Online Offline
displayUserIdentityStore	Display a user identity store registration.	Online
createOAMServer	Create an entry for an Access Manager Server configuration.	Online Offline
editOAMServer	Edit the entry for an Access Manager Server configuration.	Online Offline
deleteOAMServer	Delete the named Access Manager Server configuration.	Online Offline

Table 4–1 (Cont.) WLST Access Manager Commands

Use this command...	To...	Use with WLST...
displayOAMServer	Display Access Manager Server configuration details.	Online Offline
configurePersistentLogin	Enable or disable the Persistent Login feature.	Online
configOAMLoginPagePref	Configure the Access Manager login page user preferences.	Online
configRequestCacheType	Configure the SSO server request cache type.	Online
displayRequestCacheType	Display the SSO server request cache type entry.	Online Offline
editOssoAgent	Edit OSSO Agent configuration details.	Online Offline
deleteOssoAgent	Delete the named OSSO Agent configuration.	Online Offline
displayOssoAgent	Display OSSO Agent configuration details.	Online Offline
editWebgateAgent	Edit 10g WebGate Agent registration details.	Online Offline
deleteWebgateAgent	Delete the named 10g WebGate Agent configuration.	Online Offline
displayWebgateAgent	Display WebGate Agent configuration details.	Online Offline
exportPolicy	Export Access Manager policy data from a test (source) to an intermediate Access Manager file.	Online
importPolicy	Import Access Manager policy data from the Access Manager file specified.	Online
importPolicyDelta	Import Access Manager policy changes from the Access Manager file specified.	Online
migratePartnersToProd	Migrate partners from the source Access Manager Server to the specified target Access Manager Server.	Online
exportPartners	Export the Access Manager partners from the source to the intermediate Access Manager file specified.	Online
importPartners	Import the Access Manager partners from the intermediate Access Manager file specified.	Online
displayTopology	List the details of deployed Access Manager Servers.	Online Offline
configureOAAMPartner	Configure the Access Manager-Oracle Adaptive Access Manager basic integration.	Online
registerOIFDAPPartner	Register Identity Federation as Delegated Authentication Protocol (DAP) Partner.	Online Offline
registerOIFDAPPartnerIDPMode	Registers Identity Federation in IDP mode.	

Table 4–1 (Cont.) WLST Access Manager Commands

Use this command...	To...	Use with WLST...
registerThirdPartyTAPPartner	Registers any third party as a Trusted Authentication Protocol (TAP) Partner.	Online
disableCoexistMode	Disable the Coexist Mode.	Online
enableOamAgentCoexist	Enables Coexist Mode for the Access Manager agent (enabling the Access Manager 11g server to own the Obssocookie set by 10g WebGate).	Online
disableOamAgentCoexist	Disables Coexist Mode for the Access Manager agent (disabling the Access Manager 11g server from the Obssocookie set by 10g WebGate).	Online
editGITOValues	Edit GITO configuration parameters.	Online
editWebgate11gAgent	Edit an 11g WebGate registration.	Online Offline
deleteWebgate11gAgent	Remove an 11g WebGate Agent registration.	Online Offline
displayWebgate11gAgent	Display an 11g WebGate Agent registration.	Online Offline
displayOAMMetrics	Display metrics of Access Manager Servers.	Online Offline
updateOIMHostPort (deprecated)	Update the Oracle Identity Manager configuration when integrated with Access Manager.	Online
configureOIM (deprecated)	Creates an Agent registration specific to Oracle Identity Manager when integrated with Access Manager.	Online
updateOSSOResponseCookieConfig	Updates OSSO Proxy response cookie settings.	Online
deleteOSSOResponseCookieConfig	Deletes OSSO Proxy response cookie settings.	Online
configureAndCreateIdentityStore	Configures an identity store and external user store.	Online
configAndCreateIdStoreUsingPropFile	Configures an identity store and external user store using values defined in a file.	Online
migrateArtifacts (deprecated)	Migrates artifacts based on the specified artifact file.	Online
displaySimpleModeGlobalPassphrase	Displays the simple mode global passphrase in plain text from the system configuration.	Online
exportSelectedPartners	Exports selected Access Manager Partners to the intermediate Access Manager file specified.	Online
oamMigrate	Migrates policies, authentication stores, and user stores from OSSO, OAM10g, OpenSSO, or AM 7.1 to OAM11g.	Online
preSchemeUpgrade	Invokes the preSchemeUpgrade operation.	Online
postSchemeUpgrade	Invokes the postSchemeUpgrade operation.	Online

Table 4–1 (Cont.) WLST Access Manager Commands

Use this command...	To...	Use with WLST...
oamSetWhiteListMode	Set to true and the Access Manager Server will redirect to the URLS specified in the WhiteListURL list only.	Online
oamWhiteListURLConfig	Add, update or remove whitelist URL entries from configuration file.	Online
enableMultiDataCentreMode	Enable Multi Data Centre Mode.	Online
disableMultiDataCentreMode	Disable Multi Data Centre Mode.	Online
setMultiDataCentreClusterName	Set the Multi Data Centre Cluster name.	Online
setMultiDataCentreLogoutURLs	Set the Multi Data Centre logout URLs.	Online
addPartnerForMultiDataCentre	Add partner for Multi Data Centre.	Online
removePartnerForMultiDataCentre	Remove partner from Multi Data Centre.	Online
addOAMSSOProvider	Adds an OAM SSO provider.	Online

displayAuthZCallBackKey

Online command that allows generation and retrieval of the key used to hash the resource URL that is returned during authorization when a success/failure URL is configured for the policy.

Description

Allows retrieval of the key used to hash the resource URL during authorization if already present. If the key is not present it is created and returned. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
displayAuthZCallBackKey()
```

There are no arguments for this command.

Example

The following example displays the hash key.

```
displayAuthZCallBackKey()
```

updateCustomPages

Enables and disables custom error and login page configuration.

Description

Adds a context path and page extension to `oam-config.xml` that points to the WAR containing the custom Error and login pages:

```
<Setting Name="ssoengine" Type="htf:map">
<Setting Name="ErrorConfig" Type="htf:map">
<Setting Name="ErrorMode" Type="xsd:string">EXTERNAL</Setting>
<Setting Name="CustomPageExtension" Type="xsd:string">jsp</Setting>
<Setting Name="CustomPageContext" Type="xsd:string">/SampleApp</Setting>
</Setting>
</Setting>
```

Syntax

```
updateCustomPages(pageExtension="<fileExtension>", context="<contextPath>")
```

Argument	Definition
context	Specifies the context path to the application; for example, /SampleApp.
pageExtension	Has a default value of "jsp" but can be left blank.

Example

To enable the Custom Error page functionality, use `updateCustomPages` with the `context` and `pageExtension` parameters. This will modify the `oam-config.xml` file and enable the custom page functionality.

```
updateCustomPages(pageExtension ="jsp", context="/SampleApp")
```

To disable the Custom Error page functionality, use the command without parameters [`updateCustomPages()`]. This will undo the modifications made when the command is run with parameters.

createUserIdentityStore

Creates an identity store registration in the Access Manager system configuration.

Description

Creates an entry in the system configuration for a new user identity store registered with Access Manager. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
createUserIdentityStore(name="<Name>", principal="<Principal>",
credential="<Credential>", type="<Type>", userAttr="<userAttr>",
ldapProvider="<ldapProvider>", userSearchBase="<userSearchBase>",
ldapUrl="<ldapUrl>", isPrimary="<isPrimary>", isSystem="<isSystem>",
userIDProvider="<userIDProvider>", roleSecAdmin="<roleSecAdmin>",
roleSysMonitor="<roleSysMonitor>", roleAppAdmin="<roleAppAdmin>",
roleSysManager="<roleSysManager>", roleSecAdminGroups="<roleSecAdminGroups>",
roleSecAdminUsers="<roleSecAdminUsers>", groupSearchBase="<groupSearchBase>",
supplementaryReturnAttributes="<supplementaryReturnAttributes>",
domainHome="<domainHome>")
```

Argument	Definition
name	Mandatory. Specifies the unique name of the LDAP identity store being created. Use only upper and lower case alpha characters and numbers.
principal	Mandatory. Specifies the Principal Administrator of the LDAP identity store being created. For example, cn=Admin.
credential	Mandatory. Specifies the password of the Principal for the LDAP identity store being created.
type	Mandatory. Specifies the type of the LDAP identity store being created. For this command, the value would be LDAP.
userAttr	Mandatory. Specifies the user attributes of the LDAP identity store being created.
ldapProvider	Mandatory. Specifies the type of the LDAP identity store being created. The value might be ODSEE, AD, OID, OVD, SJS, OUD, and the like. This value is defined when a new user identity store is created using the Access Manager Administration Console and corresponds with Store Type in the user identity store.
userSearchBase	Mandatory. Specifies the node under which user data is stored in the LDAP identity store being created. For example, `cn=users`.
groupSearchBase	Mandatory. Specifies the node under which group data is stored in the LDAP identity store being created. For example, `cn=groups`.
ldapUrl	Mandatory. Specifies the URL of the server host (including port number) of the LDAP identity store being created. For example, `ldap://localhost:7001`.
isPrimary	Optional. Specifies whether the LDAP identity store being created is the primary identity store. Takes true or false as a value.
isSystem	Optional. Specifies whether the LDAP identity store being created is the system store. Takes true or false as a value.

Argument	Definition
userIDProvider	Optional. Specifies the underlying infrastructure with which to connect to the identity store. Only supported type is OracleUserRoleAPI.
roleSecAdminGroups	Optional. Specifies one or more comma-delimited groups with Access Manager Console Administrator privileges. Needed if it is a System Store in which the IsSystem property is set to true.
roleSecAdminUsers	Optional. Specifies one or more comma-delimited users with Access Manager Console Administrator privileges. Needed if it is a System Store in which the IsSystem property is set to true.
roleSecAdmin	Optional. Specifies the Security Administrator of the LDAP identity store being created.
roleSysMonitor	Optional. Specifies the System Monitor of the LDAP identity store being created.
roleAppAdmin	Optional. Specifies the Application Administrator of the LDAP identity store being created.
roleSysManager	Optional. Specifies the System Manager of the LDAP identity store being created.
supplementaryReturnAttributes	Specifies a comma-delimited list of attributes that need to be retrieved as part of the User object. For example: ORCL_USR_ENC_FIRST_NAME,ORCL_USR_ENC_LAST_NAME,USR_USRNAME,ORCL_USR_CTY_CODE,ORCL_USR_LANG_CODE_S,ORCL_USR_JROLE_ID_S,ORCL_USR_IND_ID,ORCL_USR_COMP_REL_ID,ORCL_USR_ASCII_IND,ORCL_ORA_UCM_VER,ORCL_ORA_UCM_SRVC
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere.

Example

The following example registers a new Oracle Internet Directory user identity store definition for use with Access Manager.

```
createUserIdentityStore(name="Name1", principal="Principal1",
credential="Credential1", type="Type1", userAttr="userAttr1",
ldapProvider="ldapProvider", userSearchBase="userSearchBase", ldapUrl="ldapUrl",
isPrimary="isPrimary", isSystem="isSystem", userIDProvider="userIDProvider",
roleSecAdmin="<roleSecAdmin>", roleSysMonitor="<roleSysMonitor>",
roleAppAdmin="<roleAppAdmin>", roleSysManager="<roleSysManager>",
roleSecAdminGroups="<roleSecAdminGroups>",
roleSecAdminUsers="<roleSecAdminUsers>", groupSearchBase="groupSearchBase",
supplementaryReturnAttributes="supplementaryReturnAttributes",
domainHome="domainHome1")
```

editUserIdentityStore

Online and offline command that modifies an already defined identity store registration for Access Manager.

Description

Changes one or more attributes of the user identity store registered with Access Manager. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
editUserIdentityStore(name="<Name>", [ principal="<Principal>",
credential="<Credential>", type="<Type>", userAttr="<userAttr>",
ldapProvider="<ldapProvider>", roleSecAdmin="<roleSecAdmin>",
roleSysMonitor="<roleSysMonitor>", roleSysManager="<roleSysManager>" ,
roleAppAdmin="<roleAppAdmin>", roleSecAdminGroups="<roleSecAdminGroups>",
roleSecAdminUsers="<roleSecAdminUsers>", userSearchBase="<userSearchBase>",
ldapUrl="<ldapUrl>", isPrimary="<isPrimary>", isSystem="<isSystem>",
userIDProvider="<userIDProvider>" , groupSearchBase="<groupSearchBase>",
domainHome="<domainHome>", userFilterObjectClasses="<userFilterObjectClasses>",
groupFilterObjectClasses="<groupFilterObjectClasses>",
referralPolicy="<referralPolicy>", searchTimeLimit="<searchTimeLimit>",
minConnections="<minConnections>", maxConnections="<maxConnections>",
connectionWaitTimeout="<connectionWaitTimeout>",
connectionRetryCount="<connectionRetryCount>", groupNameAttr="<groupNameAttr>",
groupCacheEnabled="<groupCacheEnabled>", groupCacheSize="<groupCacheSize>",
groupCacheTTL=<"groupCacheTTL>",
supplementaryReturnAttributes="<supplementaryReturnAttributes>" )
```

Argument	Definition
name	Mandatory. Specifies the unique name of the LDAP identity store being modified. Use only upper and lower case alpha characters and numbers.
principal	Specifies the Principal Administrator of the LDAP identity store being modified. For example, `cn=Admin`.
credential	Specifies the encrypted Password of the Principal Administrator for the LDAP identity store being modified.
type	Specifies the type of the base identity store being modified. For this command, the value would be LDAP.
userAttr	Mandatory. Specifies the user attributes of the LDAP identity store being modified.
ldapProvider	Mandatory. Specifies the LDAP type of the LDAP identity store being registered. The value might be ODSEE, AD, OID, OVD, SJS, OUD, and the like. This value is defined when a new user identity store is created using the Access Manager Administration Console and corresponds with Store Type in the user identity store.
roleSecAdminGroups	Optional. Specifies one or more comma-delimited groups with Access Manager Console Administrator privileges. Needed if it is a System Store in which the IsSystem property is set to true.
roleSecAdminUsers	Optional. Specifies one or more comma-delimited users with Access Manager Console Administrator privileges. Needed if it is a System Store in which the IsSystem property is set to true.

Argument	Definition
roleSecAdmin	Optional. Specifies the Security Administrator of the LDAP identity store being modified.
roleSysMonitor	Optional. Specifies the System Monitor of the LDAP identity store being modified.
roleAppAdmin	Optional. Specifies the Application Administrator of the LDAP identity store being modified.
roleSysManager	Optional. Specifies the System Manager of the LDAP identity store being modified.
userSearchBase	Mandatory. Specifies the node under which user data is stored in the LDAP identity store being modified. For example, `cn=users`.
groupSearchBase	Mandatory. Specifies the node under which user data is stored in the LDAP identity store being modified. For example, `cn=groups`.
ldapUrl	Mandatory. Specifies the URL of the server host (including port number) of the LDAP identity store being modified. For example, `ldap://localhost:7001`.
isPrimary	Optional. Specifies whether the LDAP identity store being modified is the primary identity store. Takes true or false as a value.
isSystem	Optional. Specifies whether the LDAP identity store being modified is the system store. Takes true or false as a value.
userIDProvider	Optional. Specifies the underlying infrastructure with which to connect to the identity store. Only supported type is OracleUserRoleAPI.
supplementaryReturnAttributes	Specifies a comma-delimited list of attributes that need to be retrieved as part of the User object. For example: ORCL_USR_ENC_FIRST_NAME,ORCL_USR_ENC_LAST_NAME,USR_USRNAME,ORCL_USR_CTY_CODE,ORCL_USR_LANG_CODE_S,ORCL_USR_JROLE_ID_S,ORCL_USR_IND_ID,ORCL_USR_COMP_REL_ID,ORCL_USR_ASCII_IND,ORCL_ORA_UCM_VER,ORCL_ORA_UCM_SRVC
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.
userFilterObjectClasses	Mandatory. Specifies a list of user filter object classes (separated by semicolon).
groupFilterObjectClasses	Specifies a list of group filter object classes (separated by semicolon).
referralPolicy	Specifies an LDAP referral policy (either "follow", "ignore" or "throw").
searchTimeLimit	Specifies the time limit in seconds for an LDAP Search operation.
minConnections	Specifies the minimum number of connections in the connection pool.
maxConnections	Specifies the maximum number of connections in the connection pool.
connectionWaitTimeout	Specifies the number of seconds to wait for obtaining a connection from the pool.
connectionRetryCount	Specifies the number of attempts to retry when establishing a connection to the identity store.
groupNameAttr	Specifies the name of the attribute to lookup the user groups. For example, `ou=people,ou=myrealm,dc=base_domain`.
groupCacheEnabled	A boolean that specifies whether to enable the LDAP group cache. Takes true or false as a value.

Argument	Definition
groupCacheSize	Specifies the number of entries in the LDAP group cache.
groupCacheTTL	Specifies the total time to live for each entry in the LDAP group cache.

Example

The following example changes the search base values for the registered identity store.

```
editUserIdentityStore(name="IdStore1", userSearchBase="cn=users",
groupSearchBase="cn=groups")
```

deleteUserIdentityStore

Online and offline command that removes an already defined identity store registration for Access Manager.

Description

Deletes the identity store registration. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
deleteUserIdentityStore(name="<name>", domainHome="<domainHome>")
```

Argument	Definition
name	Mandatory. Specifies the name of the LDAP identity store registration to be removed.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example can be used on WebSphere and deletes the registration of the named identity store. To use this command in online mode with WebLogic Server, the domainHome argument need not be specified.

```
deleteUserIdentityStore(name="identity_store", domainHome="domainHome1")
```

displayUserIdentityStore

Online command that displays user identity store registration information.

Description

Displays the information regarding the identity store registered with Access Manager. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
displayUserIdentityStore(name="<name>", domainHome="<domainHome>")
```

Argument	Definition
name	Mandatory. Specifies the name of the LDAP identity store registration to be displayed.
domainhome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere.

Example

The following example invocation for WebSphere displays registration details of the user identity store. To use this command in online mode with WebLogic, there is no need to specify the domainHome argument.

```
displayUserIdentityStore(name="ID_Store1", domainHome="domainHome1")
```

createOAMServer

Online and offline command that creates an Access Manager Server entry in the system configuration.

Description

Creates an Access Manager Server registration. Details include the host, port, registration name, Access Manager Proxy port, server ID and, optionally, the OAM Proxy shared secret. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
createOAMServer(configurationProfile="<configurationProfile>", host="<host>",
port="<port>", oamProxyPort="<0000>", oamProxyServerID="<oamProxyServerID>",
siteName="<siteName>", domainHome="<domainHome>")
```

Argument	Definition
configurationProfile	Mandatory. Specifies the Configuration Profile of the OAM Server. The profile appears under Server Instances on the System Configuration tab in the Access Manager Administration Console.
host	Mandatory. Specifies the name of the Access Manager Server host.
port	Mandatory. Specifies the listening port of the Access Manager Server host.
oamProxyPort	Mandatory. Specifies the proxy port of the Access Manager Server host.
oamProxyServerID	Mandatory. Specifies the proxy server ID of the Access Manager Server host. The Access Manager Proxy name appears under the Access Manager Proxy sub tab of the server instance in the Access Manager Administration Console.
siteName	Mandatory. Specifies the siteName/serverName for the instance.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example creates a configuration for `my_host` with listening port `15000`. The configuration entry in the Access Manager Administration Console will be `oam_server1`. The Access Manager Proxy port is 3004 and the Access Manager Proxy Server ID is `oamProxyServerID1`.

```
createOAMServer(configurationProfile="oam_server1", host="my_host",
port="15000", oamProxyPort="3004", oamProxyServerID="oamProxyServerID1",
siteName="siteName1", domainHome="domainHome1")
```

editOAMServer

Online and offline command that enables you to modify the details of an Access Manager Server registration.

Description

Modifies the specified parameter values of the registration for an Access Manager Server. Details may include the host, port, registration name, Access Manager Proxy port, server ID and, optionally, the Access Manager Proxy shared secret. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
editOAMServer(configurationProfile="<configurationProfile>", host="<host>",
port="<port>", oamProxyPort="<0000>", oamProxyServerID="<oamProxyServerID>",
siteName="<siteName>", domainHome="<domainHome>")
```

Argument	Definition
configurationProfile	Mandatory. Specifies the Configuration Profile of the Access Manager Server. The profile appears under Server Instances on the System Configuration tab in the Access Manager Administration Console.
host	Mandatory. Specifies the name of the Access Manager Server host.
port	Mandatory. Specifies the listening port of the Access Manager Server host.
oamProxyPort	Mandatory. Specifies the proxy port of the Access Manager Server host.
oamProxyServerID	Mandatory. Specifies the proxy server ID of the Access Manager Server host. The Access Manager Proxy name appears under the Access Manager Proxy sub tab of the server instance in the Access Manager Administration Console.
siteName	Mandatory. Specifies the siteName/serverName for the instance.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

You can use any of the optional attributes to change current settings. The following invocation enables you to add the Access Manager Proxy Sever ID to the configuration entry `oam_server1`.

```
editOAMServer(configurationProfile="oam_server1", host="my_host",
port="15000", oamProxyPort="3004", oamProxyServerID="oamProxyServerID1",
siteName="siteName1", domainHome="domainHome1")
```

deleteOAMServer

Online and offline command that enables you to delete the specified Access Manager Server registration.

Description

Deletes the specified Access Manager Server configuration. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
deleteOAMServer(host="<host>", port="<port>", domainHome="<domainHome>")
```

Argument	Definition
host	Mandatory. Specifies the name of the Access Manager Server host.
port	Mandatory. Specifies the listening port of the Access Manager Server host.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example enables you to delete the `oam_server1` Access Manager Server registration with listening port `15000`.

```
deleteOAMServer(host="oam_server1", port="15000", domainHome="domainHome1")
```

displayOAMServer

Online and offline command that displays registration details for the specified Access Manager Server.

Description

Displays the registration details of the specified Access Manager Server, including the host, port, registration name, Access Manager Proxy port, server ID and, optionally, the Access Manager Proxy shared secret. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
displayOAMServer(host="<host>", port="<port>", domainHome="<domainHome>")
```

Argument	Definition
host	Mandatory. Specifies the name of the Access Manager Server host.
port	Mandatory. Specifies the listening port of the Access Manager Server host.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example will list all metrics specific to the `my_host` Access Manager Server.

```
displayOAMServer(host="my_host", port="15000", domainHome="domainHome1")
```

configurePersistentLogin

Online command to enable or disable the Persistent Login feature.

Description

Enables the Persistent Login feature.

Syntax

```
configurePersistentLogin(enable="true/false",
 validityInDays="<#>", maxAuthnLevel="<#>", userAttribute="<userAttr>")
```

Argument	Definition
enable	Mandatory. Specify true or false.
validityInDays	Mandatory. Specifies the number of days that the user login will be persisted for a particular browser instance or device.
maxAuthnLevel	Mandatory. Specifies the maximum Authentication Level allowed after re-authenticating automatically through Persistent Login.
userAttr	Mandatory. Specifies the user attribute with which Persistent Login properties will be stored.

Example

The following example changes the search base values for the registered identity store.

```
configurePersistentLogin(enable="true", validityInDays="30", maxAuthnLevel="2"
 userAttribute="obPSFTID")
```

configOAMLoginPagePref

Online command that configures the Access Manager login page user preferences.

Description

Configures the Access Manager login page user preferences.

Syntax

```
configOAMLoginPagePref(persistentCookie="true", persistentCookieLifetime=14,
langPrefCookieDomain="oracle.com", langPrefOrder="serverOverrideLangPref,
oamPrefsCookie, browserAcceptLanguage, defaultLanguage",
serverOverrideLanguage="en", defaultLanguage="en",
applicationSupportedLocales="en,fr")
```

Argument	Definition
persistentCookie	Mandatory. Boolean that defines whether the OAM_LANG_PREF cookie is persistent or non-persistent. Set to true or false.
persistentCookieLifetime	Mandatory. Lifetime of the OAM_LANG_PREF cookie if persistent.
langPrefCookieDomain	Mandatory. Defines the domain of the OAM_LANG_PREF cookie.
langPrefOrder	Mandatory. Decides the order of language precedence. Must be formatted as in the syntax and example. The allowed value set is (serverOverrideLangPref,oamPrefsCookie,browserAcceptLanguage,defaultLanguage). "oamPrefsCookie, browserAcceptLanguage, serverOverrideLangPref"
serverOverrideLanguage	The server side language of Access Manager. Must be defined in language codes and selected from OAM supported languages. Default value is en.
defaultLanguage	The default language.
applicationSupportedLocales	Supported languages defined in a comma-delimited list. Setting `applicationSupportedLocales="en,fr"` insures the OAM Login page will display a list of values containing French and English. The supported language codes are documented in Table 4–2 below.

Table 4–2 Language Codes For Login Pages

Language Code	Language	Administrators
ar	Arabic	
cs	Czech	
da	Danish	
de	German	German
el	Greek	
en	English	English
es	Spanish	Spanish
fi	Finnish	

Table 4–2 (Cont.) Language Codes For Login Pages

Language Code	Language	Administrators
fr	French	French
fr-CA	Canadian French	Canadian French
he	Hebrew	
hr	Croatian	
hu	Hungarian	
it	Italian	Italian
ja	Japanese	Japanese
ko	Korean	Korean
nl	Dutch	
no	Norwegian	
pl	Polish	
pt-BR	Brazilian Portuguese	Brazilian Portuguese
pt	Portuguese	
ro	Romanian	
ru	Russian	
sk	Slovak	
sv	Swedish	
th	Thai	
tr	Turkish	
zh-CN	Simplified Chinese	Simplified Chinese
zh-TW	Traditional Chinese	Traditional Chinese

Example

```
configOAMLoginPagePref(persistentCookie="true", persistentCookieLifetime=14,
langPrefCookieDomain="oracle.com", langPrefOrder="serverOverrideLangPref,
oamPrefsCookie, browserAcceptLanguage, defaultLanguage",
serverOverrideLanguage="en", defaultLanguage="en",
applicationSupportedLocales="en,fr")
```

This next example allows an administrator to revert back to the default behavior in which no language list of values is displayed.

```
configOAMLoginPagePref(persistentCookie="true",
persistentCookieLifetime=14,langPrefCookieDomain="example.com",
langPrefOrder="serverOverrideLangPref,oamPrefsCookie,browserAcceptLanguage,
defaultLanguage",serverOverrideLanguage="",
defaultLanguage="en",applicationSupportedLocales="")
```

configRequestCacheType

Online and offline command that defines the SSO server request cache type in the system configuration.

Description

Defines the SSO server request cache type in the system configuration. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
configRequestCacheType(type="<requestCacheType>", domainHome="<domainHome>")
```

Argument	Definition
type	Mandatory. Specifies the request cache type. Takes a value of BASIC or COOKIE.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example identifies the request cache type as Cookie:

```
configRequestCacheType(type="COOKIE")
```

displayRequestCacheType

Online and offline command that displays the SSO server request cache type defined for the specified domain. The request cache type may be BASIC or COOKIE.

Description

Displays the SSO server request cache type entry defined for the specified domain. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
displayRequestCacheType(domainHome="<domainHome>")
```

Argument	Definition
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example will display the request cache type (BASIC or COOKIE) defined for the specified domain home.

```
displayRequestCacheType(domainHome="domainHome1")
```

editOssoAgent

Online and offline command that enables you to modify the details of an OpenSSO (OSSO) Agent registration in the system configuration.

Description

Modifies OSSO Agent registration details including the Site Token, Success URL, Failure URL, Home URL, Logout URL, Start Date, End Date, Administrator ID, and Administrator Info. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
editOssoAgent(agentName="AgentName", partnerId = "<partnerId>",
siteToken = "<siteToken>", siteName = "<siteName>", successUrl ="<successUrl>",
failureUrl = "<failureUrl>", homeUrl="<homeUrl>", logoutUrl="<logoutUrl>",
startDate = "<startDate>", endDate = "<endDate>", adminId = "<adminId>",
adminInfo = "<AdminInfo>", domainHome="<domainHomeName>")
```

Argument	Definition
agentName	Mandatory. Specifies the name of the OSSO Agent entry to be modified. adminId=admin Id of OSSO agent <optional> adminInfo=admin Information of OSSO agent <optional>
partnerId	Optional. Specifies the Agent Name of the OSSO agent instance.
siteToken	Optional. Specifies the Application Token used by the partner when requesting authentication.
siteName	Optional. Specifies the SiteName/ServerName for the OSSO agent instance.
successUrl	Optional. Specifies the redirect URL to be used by the OSSO Agent if authentication is successful.
failureUrl	Optional. Specifies the redirect URL to be used by the OSSO Agent if authentication fails.
homeUrl	Optional. Specifies the redirect URL to be used for the Home page after authentication.
logoutUrl	Optional. Specifies the redirect URL to be used when a user is logging out.
startDate	Optional. Specifies the first month, day, and year for which login to the application is allowed by the server.
endDate	Optional. Specifies the final month, day, and year for which login to the application is allowed by the server.
adminId	Optional. Specifies the administrator login ID for the OSSO Agent.
adminInfo	Optional. Specifies an administrator identifier for the OSSO Agent for tracking purpose.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example changes the Administrator ID and information in the registration entry for `OSSOAgent1`.

```
editOssoAgent(agentName = "OSSOAgent1", partnerId = "partnerId",
siteToken = "siteToken", siteName = "siteName", successUrl="successUrl",
failureUrl = "failureUrl", homeUrl="homeUrl", logoutUrl="logoutUrl",
startDate = "2009-12-10", endDate = "2012-12-30", adminId = "345",
adminInfo = "Agent11", domainHome="domainHome1")
```

deleteOssoAgent

Online and offline command that enables you to remove the specified OSSO Agent registration in the system configuration.

Description

Removes the specified OSSO Agent registration in the system configuration. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
deleteOssoAgent(agentName="<AgentName>", domainHome="<domainHomeName>")
```

Argument	Definition
agentName	Mandatory. Specifies the name of the OSSO Agent entry to be removed.
domainhome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example removes the OSSO Agent registration entry named `OSSOAgent1`.

```
deleteOssoAgent(agentName="OSSOAgent1", domainHome="domainHome1")
```

displayOssoAgent

Online and offline command that displays the details of the specified OSSO Agent entry in the system configuration.

Description

Displays the details of the specified OSSO Agent entry in the Access Manager Administration Console. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
displayOssoAgent(agentName="<AgentName>", domainHome="<domainHomeName>")
```

Argument	Definition
agentName	Mandatory. Specifies the name of the OSSO Agent entry to be displayed.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example displays the `OSSOAgent1` entry details.

```
displayOssoAgent(agentName="OSSOAgent1", domainHome="domainHome1")
```

editWebgateAgent

Online and offline command that enables you to modify a Webgate 10g registration entry in the system configuration.

Description

Enables you to modify a Webgate 10g registration entry in the system configuration. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
editWebgateAgent(agentName="<AgentName>",
accessClientPasswd="<accessClientPassword >",
state="<state>", preferredHost="<host>",
aaaTimeOutThreshold="<aaaTimeoutThreshold >", security="<security>",
primaryCookieDomain="<primaryCookieDomain>", maxConnections="<maxConnections>",
maxCacheElems="<maxCacheElements >", cacheTimeout="<cacheTimeOut>",
cookieSessionTime="<cookieSessionTime >", maxSessionTime="<maxSessionTime>",
idleSessionTimeout="<idleSessionTimeout >",
failoverThreshold="<failoverThreshold >", domainHome="<domainHomeName>")
```

Argument	Definition
agentName	Mandatory. Specifies the name of the WebGate Agent to be modified.
accessClientPasswd	Optional. Specifies the access client password of WebGate Agent.
state	Optional. Specifies whether the WebGate Agent is enabled or disabled with a value of either Enabled or Disabled, respectively.
preferredHost	Optional. Specifies the preferred host of the WebGate Agent. This prevents security holes that can be created if a host's identifier is not included in the Host Identifiers list. For virtual hosting, you must use the Host Identifiers feature.
aaaTimeOutThreshold	Optional. Specifies the number (in seconds) to wait for a response from the Access Manager run-time server. If this parameter is set, it is used as an application TCP/IP timeout instead of the default TCP/IP timeout. Default = -1 (default network TCP/IP timeout is used)
security	Optional. Specifies the level of transport security to and from the Access Manager run-time server. Takes as a value either open, simple, or cert.
primaryCookieDomain	Optional. Specifies the Web server domain on which the Access Manager Agent is deployed. For example, *.acompany.com*
maxConnections	Optional. Specifies the maximum number of connections that this Access Manager Agent can establish with the Access Manager Server. This number must be the same as (or greater than) the number of connections that are actually associated with this agent. Default = 1
maxCacheElems	Optional. Specifies the maximum number of elements maintained in the cache. Cache elements are URLs or Authentication Schemes. The value of this setting refers to the maximum consolidated count for elements in both of these caches. Default = 10000
cacheTimeout	Optional. Specifies the amount of time cached information remains in the Access Manager Agent cache when the information is neither used nor referenced. Default = 1800 (seconds)
cookieSessionTime	Optional. Specifies the amount of time that the ObSSOCookie persists. Default = 3600 (seconds)

Argument	Definition
maxSessionTime	Optional. Specifies the maximum amount of time in seconds that a user's authentication session is valid regardless of their activity. At the expiration of this time, the user is re-challenged for authentication. This is a forced logout. A value of 0 disables this timeout setting. Default = 3600 (seconds)
idleSessionTimeout	Specifies the location of the Domain Home. When Offline, a value is mandatory; when online, optional.
failoverThreshold	Optional. Specifies a number representing the point when this Access Manager Agent opens connections to a Secondary Access Manager Server. Default = 1
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

You can alter any or all of the settings. Use the following example to change the Agent ID, state, maximum connections, Access Manager Server timeout, primary cookie domain, cache time out, cookie session timeout, maximum session timeout, idle session timeout, and failover threshold.

```
editWebgateAgent(agentName="WebgateAgent1", accessClientPasswd="welcome1",
state="Enabled", preferredHost="141.144.168.148:2001", aaaTimeOutThreshold = "10",
security="open", primaryCookieDomain="primaryCookieDomain", maxConnections="16",
maxCacheElems="10000", cacheTimeout="1800", cookieSessionTime="3600",
maxSessionTime="24", idleSessionTimeout="3600", failoverThreshold="1",
domainHome="domainHome1")
```

deleteWebgateAgent

Online and offline command that enables you to delete a Webgate_agent registration entry in the system configuration.

Description

Removes the specified Webgate_agent registration entry from the system configuration. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
deleteWebgateAgent(agentName="<AgentName>", domainHome="<domainHomeName>")
```

Argument	Definition
agentName	Mandatory. Specifies the name of the WebGate Agent being deleted.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example removes the WebGate Agent named `WebgateAgent1`.

```
deleteWebgateAgent(agentName="WebgateAgent1", domainHome="domainHome1")
```

displayWebgateAgent

Online and offline command that displays a Webgate_agent registration entry.

Description

Displays all details of the specified Webgate_agent registration entry in the Access Manager Administration Console. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
displayWebgateAgent(agentName="<AgentName>", domainHome="<domainHomeName>")
```

Argument	Definition
agentName	Mandatory. Specifies the name of the WebGate Agent being displayed.
domainhome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example displays entry details for `WebgateAgent1`.

```
displayWebgateAgent(agentName="WebgateAgent1", domainHome="domainHome1")
```

exportPolicy

Online only command that exports Access Manager policy data from a test (source) environment to the intermediate Access Manager file specified.

Description

Exports Access Manager policy data from a test (source) environment to the intermediate Access Manager file. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
exportPolicy(pathTempOAMPolicyFile="<absoluteFilePath>")
```

Argument	Definition
pathTempOAMPolicyFile	Mandatory. Specifies the absolute path to the temporary Access Manager file.

Example

The following example specifies the path to the `tempfile.txt` file used when exporting policy data from a test (source) environment.

```
exportPolicy(pathTempOAMPolicyFile="/exampleroot/parent/tempfile.txt")
```

importPolicy

Online only command that imports the Access Manager policy data from the specified Access Manager file.

Description

Imports the Access Manager policy data from the specified Access Manager file. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
importPolicy(pathTempOAMPolicyFile="<absoluteFilePath>")
```

Argument	Definition
pathTempOAMPolicyFile	Mandatory. Specifies the absolute path to the temporary Access Manager file.

Example

The following example specifies the path to the `tempfile.txt` file used when importing policy data to a production (target) environment.

```
importPolicy(pathTempOAMPolicyFile="/exampleroot/parent/tempfile.txt")
```

importPolicyDelta

Online only command that imports the Access Manager policy changes from the specified Access Manager file.

Description

Imports the Access Manager policy changes from the specified Access Manager file. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
importPolicyDelta(pathTempOAMPolicyFile="<absoluteFilePath>")
```

Argument	Definition
pathTempOAMPolicyFile	Mandatory. Specifies the absolute path to the temporary Access Manager file.

Example

The following example specifies the path to the `tempfile_delta.txt` file used when importing changed policy data to a production (target) environment.

```
importPolicyDelta(pathTempOAMPolicyFile="/exampleroot/parent/tempfile_delta.txt")
```

migratePartnersToProd

Online only command that migrates partners from the current (source) Access Manager Server to the specified (target) Access Manager Server.

Description

Migrates partners from the current (source) Access Manager Server to the specified (target) Access Manager Server. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
migratePartnersToProd(prodServerHost="<host>", prodServerPort="<port>",
prodServerAdminUser="<user>", prodServerAdminPwd="<passwd>")
```

Argument	Definition
prodServerHost	Host name of the target Access Manager Server to which partners are to be migrated.
prodServerPort	Port of the target Access Manager Server to which partners are to be migrated.
prodServerAdminUser	Administrator of the target Access Manager Server to which partners are to be migrated.
prodServerAdminPwd	Target Access Manager Server administrator's password.

Example

The following example specifies the required information for partner migration.

```
migratePartnersToProd(prodServerHost="myhost", prodServerPort="1234",
prodServerAdminUser="weblogic", prodServerAdminPwd="welcome")
```

exportPartners

Online only command that exports Access Manager partners from the source to the Access Manager file specified.

Description

Exports the Access Manager partners from the source to the Access Manager file specified. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
exportPartners(pathTempOAMPartnerFile="<absoluteFilePath>")
```

Argument	Definition
pathTempOAMPolicyFile	Mandatory. Specifies the absolute path to the temporary Access Manager file.

Example

The following example specifies the absolute path to the Access Manager partners file.

```
exportPartners(pathTempOAMPolicyFile="/exampleroot/parent/tempfile_partners.xml")
```

importPartners

Online only command that imports Access Manager partners from the specified Access Manager file.

Description

Imports the Access Manager partners from the specified Access Manager file. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
importPartners(pathTempOAMPartnerFile="<absoluteFilePath>")
```

Argument	Definition
pathTempOAMPartnerFile	Mandatory. Specifies the path to the temporary Access Manager partner file.

Example

The following example specifies the absolute path to the Access Manager file from which the partners will be imported.

```
importPartners(pathTempOAMPolicyFile="/exampleroot/parent/tempfile_partners.xml")
```

displayTopology

Online and offline command that displays information about all Access Manager Servers in a deployment.

Description

Lists the topology of deployed Access Manager Servers.

Syntax

```
displayTopology(domainHome="<domainHomeName>")
```

Argument	Definition
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example lists the details of all deployed Access Manager Servers in the specified domain home.

```
displayTopology(domainHome="domainHome1")
```

configureOAAMPartner

Online only command that configures the basic integration of Access Manager and Oracle Adaptive Access Manager (OAAM).

Description

Configures the basic integration of Access Manager and OAAM. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
configureOAAMPartner(dataSourceName="<dataSourceName>", hostName="<hostName>",
port="<port>", serviceName="<serviceName>", userName="<userName>",
passWord="<passWord>", maxConnectionSize="<maxConnectionSize>",
maxPoolSize="<maxPoolSize>", serverName="<serverName>")
```

Argument	Definition
dataSourceName	Mandatory. Specifies the name of the data source to be created.
hostName	Mandatory. Specifies the name of the database host.
port	Mandatory. Specifies the database port number.
serviceName	Mandatory. Specifies the database service name.
userName	Mandatory. Specifies the OAAM schema name.
passWord	Mandatory. Specifies the OAAM schema password.
maxConnectionSize	Optional. Specifies the maximum connection reserve time out size.
maxPoolSize	Optional. Specifies the maximum size for the connection pool.
serverName	Optional. Specifies the target server for the data source.

Example

The following example configures a basic integration for Access Manager and OAAM.

```
configureOAAMPartner(dataSourceName="MyOAAMDS", hostName="host.example.com",
port="1521", serviceName="sevice1", userName="username", passWord="password",
maxConnectionSize=None, maxPoolSize=None, serverName="oam_server1")
```

registerOIFDAPPartner

Online and offline command that registers Oracle Access Management Identity Federation (Identity Federation) as a Delegated Authentication Protocol (DAP) Partner.

Description

Registers Identity Federation as Delegated Authentication Protocol (DAP) Partner. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
registerOIFDAPPartner(keystoreLocation="/scratch/keystore"
logoutURL="http://<oifhost>:<oifport>/fed/user/splooam11g?
 doneURL=http(s)://<oamhost>:<oamport>/oam/server/pages/logout.jsp",
rolloverTime="nnn")
```

Argument	Definition
keystoreLocation	Mandatory. Specifies the location of the Keystore file (generated at the Identity Federation Server).
logoutURL	Mandatory. Specifies the logout URL for the Identity Federation server.
rolloverTime	Optional. Specifies the amount of time in seconds for which the keys used to encrypt/decrypt SASSO tokens can be rolled over.

Example

The following example illustrates the use of the parameters.

```
registerOIFDAPPartner(keystoreLocation="/scratch/keystore",
logoutURL="http(s)://oif.mycompany.com:1234/fed/user/splooam11g?
doneURL=http(s)://oam.mycompany.com:5678/oam/server/pages/logout.jsp",
rolloverTime="500")
```

registerOIFDAPPartnerIDPMode

Online and offline command that registers Identity Federation as a Delegated Authentication Protocol (DAP) Partner in IDP Mode.

Description

Registers Identity Federation as Delegated Authentication Protocol (DAP) Partner in IDP Mode. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
registerOIFDAPPartnerIDPMode(logoutURL="http://<oifhost>:<oifport>/fed/user/sploos
so?doneURL=http://<oamhost>:<oamport>/ngam/server/pages/logout.jsp")
```

Argument	Definition
logoutURL	Mandatory. Specifies the logout URL for the Identity Federation server.

Example

The following example illustrates the use of the logout URL parameter.

```
registerOIFDAPPartner(
logoutURL="http://oif.oracle.com:1234/fed/user/sploosso?
 doneURL=http://oam.oracle.com:5678/ngam/server/pages/logout.jsp")
```

registerThirdPartyTAPPartner

Registers any third party as a Trusted Authentication Protocol (TAP) Partner.

Description

Registers any third party as a Trusted Authentication Protocol (TAP) Partner.

Syntax

```
registerThirdPartyTAPPartner(partnerName="ThirdPartyTAPPartner",
keystoreLocation="/scratch/DAPKeyStore/mykeystore.jks",
password="test", tapTokenVersion="v2.0", tapScheme="TAPScheme",
tapRedirectUrl="http://thirdpartyserverhost:port/loginPage.jsp")
```

Argument	Definition
partnerName	Mandatory. Specifies the name of the partner. Can be any name used to identify the third party partner.
keystoreLocation	Mandatory. Specifies the location of the keystore file.
password	Mandatory. Specifies the password for the keystore file.
tapTokenVersion	Mandatory. Specifies the version of the Trusted Authentication Protocol.
tapScheme	Optional. Specifies the TAPScheme name used to protect the resource - TAPScheme, out of the box.
tapRedirectUrl	Optional. Specifies the TAP challenge URL to which the credential collector will be redirected.

Example

The following example illustrates the use of the parameters.

```
registerThirdPartyTAPPartner(partnerName = "ThirdPartyTAPPartner",
keystoreLocation="/scratch/DAPKeyStore/mykeystore.jks",
password="test", tapTokenVersion="v2.0", tapScheme="TAPScheme",
tapRedirectUrl="http://thirdpartyserverhost:port/loginPage.jsp")
```

disableCoexistMode

Online command that disables Coexist Mode.

Description

Disables Coexist Mode. The scope of this command is an instance only; the scope is not an argument. There are no arguments for this command.

Syntax

```
disableCoexistMode()
```

Example

The following example disables Coexist Mode.

```
disableCoexistMode()
```

enableOamAgentCoexist

Enables Coexist Mode for the Access Manager agent (enabling the Access Manager 11g server to own the Obssocookie set by 10g WebGate).

Description

Enables Coexist Mode for the Access Manager agent. The scope of this command is an instance only; the scope is not an argument. There are no arguments for this command.

Syntax

```
enableOamAgentCoexist()
```

Example

The following example enables the Coexist Mode.

```
enableOamAgentCoexist
```

disableOamAgentCoexist

Disables Coexist Mode for the Access Manager agent.

Description

Disables the Coexist Mode for the Access Manager agent. The scope of this command is an instance only; the scope is not an argument. There are no arguments for this command.

Syntax

```
disableOamAgentCoexist()
```

Example

The following invocation enables the Coexist Mode.

```
disableOamAgentCoexist
```

editGITOValues

Online and offline command that edits GITO configuration parameters.

Description

Edits GITO configuration parameters. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
editGITOValues(gitoEnabled="true", gitoCookieDomain=".abc.com",
gitoCookieName="ABC", gitoVersion="v1.0", gitoTimeout="20",
gitoSecureCookieEnabled="false", domainHome="/abc/def/ijk")
```

Argument	Definition
gitoEnabled	Allows (or denies) user to set GITO enabled property. Takes a value of true or false.
gitoCookieDomain	Mandatory. Specifies the GITO cookie domain.
gitoCookieName	Optional. Specifies the cookie name.
gitoVersion	Optional. Specifies the GITO version. Takes ONLY v1.0 or v3.0.
gitoTimeout	Optional. Specifies the GITO timeout value.
gitoSecureCookieEnabled	Optional. Enables the GITO cookie enabled property. Takes a value of true or false.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example edits the GITO configuration parameters.

```
editGITOValues(gitoEnabled="true", gitoCookieDomain=".abc.com",
gitoCookieName="ABC", gitoVersion="v1.0", gitoTimeout="20",
gitoSecureCookieEnabled="false", domainHome="/abc/def/ijk")
```

editWebgate11gAgent

Online and offline command that edits an 11g Webgate_entry registration in the system configuration.

Description

Edits an 11g Webgate_entry registration in the system configuration. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
editWebgate11gAgent(agentName="<AgentName>",
accessClientPasswd="<accessClientPassword >",
state="<state>", preferredHost="<host>",
aaaTimeoutThreshold="<aaaTimeOutThreshold>", security="<security>",
logOutUrls="<logOutUrls>", maxConnections="<maxConnections>",
maxCacheElems="<maxCacheElements>", cacheTimeout="<cacheTimeOut>",
logoutCallbackUrl="<logoutCallbackUrl >",
maxSessionTime="<maxSessionTime>", logoutRedirectUrl="<logoutRedirectUrl >",
failoverThreshold="<failoverThreshold>",
tokenValidityPeriod="<tokenValidityPeriod>",
logoutTargetUrlParamName="<logoutTargetUrlParamName>", domainHome="<domainHome>",
allowManagementOperations="<allowManagementOperations>",
allowTokenScopeOperations="<allowTokenScopeOperations>",
allowMasterTokenRetrieval="<allowMasterTokenRetrieval>",
allowCredentialCollectorOperations="<allowCredentialCollectorOperations>")
```

Argument	Definition
agentName	Mandatory. Specifies the name of the 11g WebGate Agent to be modified.
accessClientPasswd	Optional. Specifies the unique client password for this WebGate Agent.
state	Optional. Specifies whether the WebGate Agent is enabled or disabled with a value of either Enabled or Disabled, respectively.
preferredHost	Optional. Specifies the preferred host of the WebGate Agent. This prevents security holes that can be created if a host's identifier is not included in the Host Identifiers list. For virtual hosting, you must use the Host Identifiers feature.
aaaTimeoutThreshold	Optional. Specifies the number (in seconds) to wait for a response from the Access Manager run-time server. If this parameter is set, it is used as an application TCP/IP timeout instead of the default TCP/IP timeout. Default = -1 (default network TCP/IP timeout is used)
security	Optional. Specifies the level of transport security to and from the Access Manager run-time server. Takes as a value either open, simple, or cert.
logOutUrls	List of URLS that trigger the logout handler, which removes the ObSSOCookie.
maxConnections	Optional. Specifies the maximum number of connections that this Access Manager Agent can establish with the Access Manager Server. This number must be the same as (or greater than) the number of connections that are actually associated with this agent. Default = 1

Argument	Definition
maxCacheElems	Optional. Specifies the maximum number of elements maintained in the cache. Cache elements are URLs or Authentication Schemes. The value of this setting refers to the maximum consolidated count for elements in both of these caches. Default = 10000
cacheTimeout	Optional. Specifies the amount of time cached information remains in the Access Manager Agent cache when the information is neither used nor referenced. Default = 1800 (seconds)
logoutCallbackUrl	The URL to oam_logout_success, which clears cookies during the call back. By default, this is based on the Agent base URL supplied during agent registration. For example: `http://<host>:<port>`
maxSessionTime	Optional. Specifies the maximum amount of time in seconds that a user's authentication session is valid regardless of their activity. At the expiration of this time, the user is re-challenged for authentication. This is a forced logout. A value of 0 disables this timeout setting. Default = 3600 (seconds)
logoutRedirectUrl	Optional. Specifies the URL (absolute path) to the central logout page (logout.html). By default, this is based on the Access Manager Administration Console host name with a default port of 14200.
failoverThreshold	Optional. Specifies a number representing the point when this Access Manager Agent opens connections to a Secondary Access Manager Server. Default = 1
tokenValidityPeriod	Optional. Specifies the amount of time in seconds that a user's authentication session remains valid without accessing any Access Manager Agent protected resources.
logoutTargetUrlParamName	Optional. The value for this is the Logout Target URL to be invoked on logout and configured at the OPSS level.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.
allowManagementOperations	Optional. Specifies the Set the flag for Allow Management Operations
allowTokenScopeOperations	Optional. Specifies the Set the flag for Allow Token Scope Operations
idleSessionTimeout	Optional. Specifies the
allowMasterTokenRetrieval	Set flag for Allow Master Token Retrieval
allowCredentialCollectorOperations	Set flag for Allow Credential Collector Operations

Example

The following example uses all mandatory and optional parameters.

```
editWebgate11gAgent(agentName="WebgateAgent1", accessClientPasswd="welcome1",
state="Enabled", preferredHost="141.144.168.148:2001", aaaTimeoutThreshold="10",
security="open", logOutUrls="http://host1.oracle.com:1234", maxConnections = "16",
maxCacheElems="10000", cacheTimeout="1800",
logoutCallbackUrl="http://host2.oracle.com:1234",
maxSessionTime="24", logoutRedirectUrl="logoutRedirectUrl",
failoverThreshold="1", tokenValidityPeriod="tokenValidityPeriod",
logoutTargetUrlParamName="logoutTargetUrl", domainHome="domainHome1",
allowManagementOperations="false", allowTokenScopeOperations="false",
```

```
allowMasterTokenRetrieval="false", allowCredentialCollectorOperations="false")
```

deleteWebgate11gAgent

Online and offline command that enables you to remove an 11g Webgate_agent entry in the system configuration.

Description

Removes an 11g Webgate_agent entry in the system configuration. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
deleteWebgate11gAgent(agentName="<AgentName>", domainHome="<domainHomeName>")
```

Argument	Definition
agentName	Mandatory. Specifies the name of the 11g WebGate Agent to be removed.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example removes the 11g Webgate_agent entry named `my_11gWebGate`.

```
deleteWebgate11gAgent(agentName="my_11gWebGate", domainHome="domainHome1")
```

displayWebgate11gAgent

Online and offline command that enables you to display an 11g Webgate_agent registration entry.

Description

Displays an 11*g* WebGate Agent registration entry. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
displayWebgate11gAgent(agentName="<AgentName>", domainHome="<domainHomeName>")
```

Argument	Definition
agentName	Mandatory. Specifies the name of the 11g WebGate Agent to be modified.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example displays the WebGate Agent named `my_11gWebGate`:

```
displayWebgate11gAgent(agentName="my_11gWebGate", domainHome="domainHome1")
```

displayOAMMetrics

Online and offline command that enables the display of metrics for Access Manager Servers.

Description

Enables the display of metrics for Access Manager Servers. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
displayOAMMetrics(domainHome="<domainHomeName>")
```

Argument	Definition
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example displays the metrics for Access Manager Servers in the specified domain.

```
displayOAMMetrics(domainHome="domainHome1")
```

updateOIMHostPort (deprecated)

DEPRECATED - Online only command that updates the Oracle Identity Manager configuration when integrated with Access Manager.

Description

Updates the Identity Manager configuration in the system configuration. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
updateOIMHostPort(hostName="<host name>", port="<port number>",
secureProtocol="true")
```

Argument	Definition
hostName	Name of the Identity Manager host.
port	Port of the Identity Manager host.
secureProtocol	Takes a value of true or false depending on whether communication is through HTTP or HTTPS.

Example

The following example illustrates this command.

```
updateOIMHostPort(hostName="OIM.oracle.com", port="7777", secureProtocol="true")
```

configureOIM (deprecated)

DEPRECATED - Online only command that registers an agent profile specific to Oracle Identity Manager when integrated with Access Manager.

Description

Creates an Agent profile specific to Oracle Identity Manager when integrated with Access Manager. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
configureOIM(oimHost="<OIM host>", oimPort="<port>",
oimSecureProtocolEnabled="true | false", oimAccessGatePwd="<AccessGatePassword>",
oimCookieDomain="<OIMCookieDomain>", oimWgId="<OIMWebgateID>",
oimWgVersion="<OIMWebgateVersion>")
```

Argument	Definition
oimHost	Name of the Oracle Identity Manager host. In the case of EDG, the front ending LBR hostname of the OIM Cluster.
oimPort	Port of the Oracle Identity Manager Managed Server. In the case of EDG, the front ending LBR port of the OIM Managed Server Cluster.
oimSecureProtocolEnabled	Takes a value of true or false depending on whether communication is through HTTP or HTTPS.
oimAccessGatePwd	If provided, the agent password for Open mode.
oimCookieDomain	Domain in which the cookie is to be set .
oimWgId	Agent registration name.
oimWgVersion	Possible values are 10g or 11g. If not provided, default is 10*g*.

Example

The following example illustrates this command.

```
configureOIM(oimHost="oracle.com", oimPort="7777",
oimSecureProtocolEnabled="true",
oimAccessGatePwd = "welcome", oimCookieDomain = "domain1",
oimWgId="<OIM Webgate ID>", oimWgVersion="10g")
```

updateOSSOResponseCookieConfig

Online and offline command that updates the OSSO Proxy response cookie settings.

Description

Updates OSSO Proxy response cookie settings. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
updateOSSOResponseCookieConfig(cookieName="<cookieName>",
cookieMaxAge="<cookie age in minutes>", isSecureCookie="true | false",
cookieDomain="<domain of the cookie>", domainHome="<domainHomeName>")
```

Argument	Definition
cookieName	Optional. Name of the cookie for which settings are updated. If not specified, the global setting is updated.
cookieMaxAge	Maximum age of a cookie in minutes. A negative value sets a session cookie.
isSecureCookie	Boolean flag that specifies if cookie should be secure (sent over SSL channel).
cookieDomain	The domain of the cookie.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example illustrates this command.

```
updateOSSOResponseCookieConfig(cookieName = "ORASSO_AUTH_HINT",
cookieMaxAge = "525600", isSecureCookie = "false",
cookieDomain=".example.com", domainHome = "<domain_home>")
```

deleteOSSOResponseCookieConfig

Online and offline command that deletes the OSSO Proxy response cookie settings in the system configuration.

Description

Deletes the OSSO Proxy response cookie settings. The scope of this command is an instance only; the scope is not an argument.

Syntax

```
deleteOSSOResponseCookieConfig(cookieName="<cookieName>",
domainHome="<domainHomeName>")
```

Argument	Definition
cookieName	Mandatory. Name of the cookie for which settings are being deleted. The global cookie setting cannot be deleted.
domainHome	Specifies the location for the Weblogic Server OR Cell Path for WebSphere. This parameter is mandatory for WebSphere. When Offline, a value is mandatory; when online, optional.

Example

The following example illustrates this command.

```
deleteOSSOResponseCookieConfig(cookieName="ORASSO_AUTH_HINT",
domainHome = "<domain_home>")
```

configureAndCreateIdentityStore

Configures the identity store and external user store.

Description

Configures the identity store and external user store using the values supplied.

Syntax

```
configureOIM(oimHost="<OIM host>", oimPort="<port>",
oimSecureProtocolEnabled="true | false", oimAccessGatePwd="<AccessGatePassword>",
oimCookieDomain="<OIMCookieDomain>", oimWgId="<OIMWebgateID>",
oimWgVersion="<OIMWebgateVersion>"), nameOfIdStore="<nameOfIdStore>",
idStoreSecurityCredential="<idStoreSecurityCredential>",
userSearchBase="<userSearchBase>", ldapUrl="<ldapUrl>",
groupSearchBase="<groupSearchBase>", securityPrincipal="<securityPrincipal>",
idStoreType="<idStoreType>", ldapProvider="<ldapProvider>",
isPrimary="<isPrimary>", userIDProvider="<userIDProvider>",
userNameAttr="<userNameAttr>"
```

Argument	Definition
oimHost	Name of the Oracle Identity Manager host. In the case of EDG, the front ending LBR hostname of the OIM Cluster.
oimPort	Port of the Oracle Identity Manager Managed Server. In the case of EDG, the front ending LBR port of the OIM Managed Server Cluster.
oimSecureProtocolEnabled	Takes a value of true or false depending on whether communication is through HTTP or HTTPS.
oimAccessGatePwd	If provided, the agent password for Open mode.
oimCookieDomain	Domain in which the cookie is to be set .
oimWgId	Agent registration name.
oimWgVersion	Possible values are 10g or 11g. If not provided, default is 10*g*.
nameOfIdStore	Mandatory. Specifies the name of the LDAP ID store to be created.
idStoreSecurityCredential	Mandatory. Specifies the password of the Principal for the LDAP identity store being created.
userSearchBase	Mandatory. Specifies the node under which user data is stored in the LDAP identity store being created.
ldapUrl	Mandatory. Specifies the URL for the LDAP host (including port number) of the LDAP identity store being created.
groupSearchBase	Mandatory. Specifies the node under which group data is stored in the LDAP identity store being created.
securityPrincipal	Mandatory. Specifies the Principal Administrator of the LDAP identity store being created.
idStoreType	Mandatory. Specifies the type of the LDAP identity store being created.
ldapProvider	Specifies the LDAP Provider type of the store being created.
isPrimary	Optional. Specifies whether the LDAP identity store being registered is the primary identity store. Takes true or false as a value.

Argument	Definition
userIDProvider	Specifies the user Identity Provider for the store being created.
userNameAttr	Mandatory. Specifies the user attributes for the store.

Example

The following example illustrates this command.

```
configureOIM(oimHost="oracle.com", oimPort="7777",
oimSecureProtocolEnabled="true",
oimAccessGatePwd = "welcome", oimCookieDomain = "domain1",
oimWgId="<OIM Webgate ID>", oimWgVersion="10g"
nameOfIdStore="nameOfIdStore",
idStoreSecurityCredential="idStoreSecurityCredential",
userSearchBase="userSearchBase", ldapUrl="ldapUrl",
groupSearchBase="groupSearchBase", securityPrincipal="securityPrincipal",
idStoreType="idStoreType", ldapProvider="ldapProvider", isPrimary="true",
userIDProvider="userIDProvider", userNameAttr="userNameAttr")
```

configAndCreateIdStoreUsingPropFile

Configures the identity store and external user store using the values supplied in a properties file.

Description

Configures the identity store and external user store using the values supplied in the specified properties file.

Syntax

```
configAndCreateIdStoreUsingPropFile(path="<path_of_property_file>")
```

Argument	Definition
path	Path to the property file in which the values are defined.

Example

The following example illustrates this command.

```
configAndCreateIdStoreUsingPropFile(path="/prop_file_directory/values.properties")
```

migrateArtifacts (deprecated)

DEPRECATED - Migrates artifacts.

Description

Migrates artifacts based on the values defined in the input artifact file.

Syntax

```
migrateArtifacts(path="<path_to_artifacts_file>", password="<password>",
type="OutOfPlace|InPlace", isIncremental="true|false")
```

Argument	Definition
path	Location of the artifacts file
password	Password used while generating original artifacts.
type	Boolean that defines the type of migration and takes as a value InPlace or OutOfPlace
isIncremental	Boolean that takes a value of true or false. If true, an incremental upgrade is done.

Example

The following example illustrates this command.

```
migrateArtifacts(path="/exampleroot/parent/t", password="welcome",
type="InPlace", isIncremental="false")
```

displaySimpleModeGlobalPassphrase

Displays the simple mode global passphrase defined in the system configuration in plain text.

Description

Online only command that displays the simple mode global passphrase in plain text. There are no arguments for this command.

Syntax

```
displaySimpleModeGlobalPassphrase()
```

Example

The following example illustrates this command.

```
displaySimpleModeGlobalPassphrase()
```

exportSelectedPartners

Exports selected Access Manager Partners to the specified Access Manager file.

Description

Exports selected Access Manager Partners to the specified Access Manager file specified.

Syntax

```
exportSelectedPartners(pathTempOAMPartnerFile="<absoluteFilePath>",
partnersNameList="<comma_separated_partner_names>")
```

Argument	Definition
pathTempOAMPartnerFile	Mandatory. The location of the file to which the information will be exported.
partnersNameList	Mandatory. Specifies a comma separated list of partner ids being exported.

Example

The following example illustrates this command.

```
exportSelectedPartners(pathTempOAMPartnerFile="/exampleroot/parent/tempfile.extn"
partnersNameList="partner1,partner2")
```

oamMigrate

Online only command that migrates policies, authentication stores, and user stores from OSSO, OAM10g, OpenSSO, or AM 7.1 to OAM11g.

Description

Invokes the `beginMigrate` operation of the migration framework mbean.

Syntax

```
oamMigrate(oamMigrateType=<migrationType>,
pathMigrationPropertiesFile="<absoluteFilePath>")
```

Argument	Definition
oamMigrateType	Mandatory. Specifies the type of migration being done. Takes one of the following as a value: OSSO | OpenSSO | OAM10g NOTE: OpenSSO applies to both SAML 7.1 and OpenSSO.
pathMigrationPropertiesFile	Mandatory. Specifies the path to the file from which the necessary artifacts for migration are read.

Example

The following example illustrates this command.

```
oamMigrate(oamMigrateType=OSSO,
pathMigrationPropertiesFile="/middlewarehome/oam-migrate.properties")
```

preSchemeUpgrade

Online only command that invokes the preSchemeUpgrade operation.

Description

Invokes the preSchemeUpgrade operation.

Syntax

```
preSchemeUpgrade
(pathUpgradePropertiesFile="/middlewarehome/oam-upgrade.properties")
```

Argument	Definition
pathUpgradePropertiesFile	Mandatory. Specifies the path to the file from which the necessary system properties for upgrade are read.

Example

The following example illustrates this command.

```
preSchemeUpgrade(pathUpgradePropertiesFile="/exampleroot/parent/tempfile.extn")
```

postSchemeUpgrade

Invokes the postSchemeUpgrade operation.

Description

Invokes the postSchemeUpgrade operation.

Syntax

```
postSchemeUpgrade
(pathUpgradePropertiesFile="/middlewarehome/oam-upgrade.properties")
```

Argument	Definition
pathUpgradePropertiesFile	Mandatory. Specifies the path to the file from which the necessary system properties for upgrade are read.

Example

The following example illustrates this command.

```
postSchemeUpgrade(pathUpgradePropertiesFile="/exampleroot/parent/tempfile.extn")
```

oamSetWhiteListMode

Sets the `oamSetWhiteListMode` to true or false.

Description

Sets the `oamSetWhiteListMode` to true or false. If true, Access Manager redirects to the last URL requested by the consuming application only if it is configured as a white-list URL.

Syntax

```
oamSetWhiteListMode(oamWhiteListMode="true|false")
```

Argument	Definition
oamWhiteListMode	Mandatory. Enables the Access Manager white list mode.

Example

The following example illustrates this command.

```
oamSetWhiteListMode(oamWhiteListMode="true")
```

oamWhiteListURLConfig

Add, update or remove whitelist URL entries from the specified file.

Description

Add, update or remove whitelist URL entries from the specified file.

This command allows you to enter whitelist URL values having wildcard port/host into the WhiteList config map.

In the value field, if host/port is specified using wildcard characters (* symbol) then all the host/port belonging to that particular format will be allowed.

On adding the * symbol, the match will be made for the WhiteList URL based on wild card comparison mechanism.

Syntax

```
oamWhiteListURLConfig(Name="xyz", Value="http://xyz.com:1234",
Operation="Remove|Update")
```

Argument	Definition
Name	Mandatory. A valid string representing the name (key) for this entry.
Value	Mandatory. A valid URL in the <protocol>://<host>:<port> format. If the port is not specified, default HTTP/HTTPS ports are assigned accordingly.
Operation	Mandatory. Takes as a value Update or Remove. Not case sensitive.

Example

The following example illustrates this command:

```
oamWhiteListURLConfig(Name="xyz", Value="http://xyz.com:1234", Operation="Update")
```

The following example illustrates this command using wildcards for Whitelist ports:

```
oamWhiteListURLConfig(Name="xyz", Value="http://xyz.com:*", Operation="Update")
oamWhiteListURLConfig(Name="xyz", Value="http://xyz.com:*", Operation="Remove")
```

The following examples illustrates this command when host/port is specified using wild card characters in value field:

```
oamWhiteListURLConfig(Name="xyz", Value="http://*.com:7777", Operation="Update")
```

The above command will allow URL's such as `http://xyz.com:7777`, `http://abc.com:7777` and so on for redirection.

```
oamWhiteListURLConfig(Name="xyz", Value="http://xyz.com:*", Operation="Update")
```

The above command will allow URL's such as `http://xyz.com:8000`, `http://abc.com:4040` and so on for redirection.

enableMultiDataCentreMode

Online only command to enable Multi Data Center Mode.

Description

Enables Multi Data Center Mode.

Syntax

```
enableMultiDataCentreMode(propfile="<absoluteFilePath>")
```

Argument	Definition
propFile	Mandatory. Specifies the absolute path to a file from which the properties to enable the Multi Data Center are read.

Example

The following example illustrates this command.

```
enableMultiDataCentre(propfile="/middlewarehome/oamMDCProperty.properties")
```

disableMultiDataCentreMode

Online only command to disable Multi Data Center Mode.

Description

Disables Multi Data Center Mode. This command has no arguments.

Syntax

```
disableMultiDataCentreMode()
```

Example

The following example illustrates this command.

```
disableMultiDataCentreMode()
```

setMultiDataCentreClusterName

Sets the Multi Data Center cluster name.

Description

Sets the Multi Data Center cluster name.

Syntax

```
setMultiDataCentreClusterName(clusterName="MyCluster")
```

Argument	Definition
clusterName	Mandatory. Specifies the name of the cluster.

Example

The following example illustrates this command.

```
postSchemeUpgrade(clusterName="MyCluster")
```

setMultiDataCentreLogoutURLs

Sets the Multi Data Center Partner logout URLs.

Description

Sets the Multi Data Center Partner logout URLs.

Syntax

```
setMultiDataCentreLogoutURLs
(logoutURLs="http://<host>:<port>/logout.jsp,http://<host>:<port>/logout.jsp")
```

Argument	Definition
logoutURLs	Mandatory. Specify a comma separated list of Multi Data Center Partner logout URLs.

Example

The following example illustrates this command.

```
setMultiDataCentreLogoutURLs(logoutURLs="http://localhost:6666/logout.jsp,http://l
ocalhost:8888/logout.jsp")
```

updateMultiDataCentreLogoutURLs

Updates the Multi Data Center Partner logout URLs.

Description

Updates the Multi Data Center Partner logout URLs.

Syntax

```
updateMultiDataCentreLogoutURLs
(logoutURLs="http://<host>:<port>/logout.jsp,http://<host>:<port>/logout.jsp")
```

Argument	Definition
logoutURLs	Mandatory. Specify a comma separated list of Multi Data Center Partner logout URLs.

Example

The following example illustrates this command.

```
updateMultiDataCentreLogoutURLs(logoutURLs="http://localhost:7777/logout.jsp,http:
//localhost:9999/logout.jsp")
```

addPartnerForMultiDataCentre

Online command that adds a partner to a Multi Data Center.

Description

Adds a partner to a Multi Data Center. This command is supported only in online mode and adds one partner at a time.

Syntax

```
addPartnerForMultiDataCentre(propfile="<absoluteFilePath>")
```

Argument	Definition
propFile	Mandatory. Specifies the absolute path to a file that contains the agent information.

Example

The following example illustrates this command.

```
addPartnerForMultiDataCentre(propfile="/middlewarehome/partnerInfo.properties")
```

removePartnerForMultiDataCentre

Removes a partner from Multi Data Center.

Description

Removes a partner from Multi Data Center. This command is supported only in online mode and removes one partner at a time.

Syntax

```
removePartnerForMultiDataCentre(webgateid="<webgateId")
```

Argument	Definition
webgateid	Mandatory. Specifies the ID of the partner to be deleted.

Example

The following example illustrates this command.

```
removePartnerForMultiDataCentre(webgateid="IAMSuite")
```

addOAMSSOProvider

Online command that adds an Access Manager SSO provider with the given login URI, logout URI, and auto-login URI.

Description

This command modifies the domain jps-config.xml by adding an Access Manager SSO service instance with the required properties. In the event of an error, the command returns a WLSTException.

Syntax

```
addOAMSSOProvider(loginuri, logouturi, autologinuri)
```

Argument	Definition
loginuri	Specifies the URI of the login page. Required.
logouturi	Specifies the URI of the logout page. Optional. If unspecified, defaults to logouturi=NONE. Set to "" to ensure that ADF security calls the OPSS logout service, which uses the implementation of the class OAMSSOServiceImpl to clear the cookie ObSSOCookie. An ADF-secured web application that would like to clear cookies without logging out the user should use this setting.
autologinuri	Specifies the URI of the autologin page. Optional. If unspecified, it defaults to autologin=NONE.

Example

The following example illustrates this command.

```
addOAMSSOProvider(loginuri="/${app.context}/adfAuthentication",
 logouturi="/oamsso/logout.html", autologinuri="/example.cgi")
```

5

Identity Federation WLST Commands

This chapter provides descriptions of custom WebLogic Scripting Tool (WLST) commands for Oracle Access Management Identity Federation (Identity Federation), including command syntax, arguments and examples.

The Identity Federation WLST commands are organized into two categories. The following sections list the Identity Federation WLST commands by category and contain links to the command reference details.

- Identity Federation Commands
- Advanced Identity Federation Commands

Note: Identity Federation WLST commands take attributes specified as key-value pairs or only the value; Oracle Access Management Access Manager takes only key-value pairs. Thus, WLST examples in this document might be defined in either manner. This WLST example uses key-value pairs.

```
setIdPPartnerAttributeProfileEntry(attrProfileID="openid-idp
-attribute-profile",
messageAttributeName="http://axschema.org/namePerson",
oamSessionAttributeName="name", requestFromIdP="true")
```

5.1 Identity Federation Commands

Use the WLST commands listed in Table 5–1 to configure federation partners and partner profiles.

Note: The Identity Federation command definitions begin with "addWSFed11IdPFederationPartner" on page 5-8.

Table 5–1 WLST Commands for Identity Federation

Use this command...	To...	Use with WLST...
addWSFed11IdPFederationPartner	Create a WS-Fed 1.1 IdP partner.	Online
addWSFed11SPFederationPartner	Create a WS-Fed 1.1 SP partner.	Online
addOpenID20IdPFederationPartner	Create an OpenID 2.0 IdP partner.	Online
addOpenID20SPFederationPartner	Create an OpenID 2.0 SP partner.	Online

Table 5–1 (Cont.) WLST Commands for Identity Federation

Use this command...	To...	Use with WLST...
addOpenID20GoogleIdPFederationPartner	Create a Google OpenID 2.0 IdP partner.	Online
addOpenID20YahooIdPFederationPartner	Create a Yahoo OpenID 2.0 IdP partner.	Online
addSAML11IdPFederationPartner	Create an IdP federation partner, including metadata, under the SAML 1.1 protocol.	Online
addSAML11SPFederationPartner	Create an SP federation partner, including metadata, under the SAML 1.1 protocol.	Online
addSAML20IdPFederationPartner	Create an IdP federation partner under the SAML 2.0 protocol.	Online
addSAML20SPFederationPartner	Create an SP federation partner under the SAML 2.0 protocol.	Online
addSAML20IdPFederationPartnerWithoutMetadata	Create an IdP federation partner under the SAML 2.0 protocol without importing metadata.	Online
addSAML20SPFederationPartnerWithoutMetadata	Create an SP federation partner under the SAML 2.0 protocol without importing metadata.	Online
configureIdPPartnerAttributeProfile	Configure an IdP partner attribute profile to specify whether incoming attributes that are not part of the profile should be ignored.	Online
configureSAML20Logout	Configure global federation logout for a SAML 2.0 federation partner.	Online
configureSAMLBinding	Configure the preferred binding for a SAML federation partner.	Online
configureUserSelfRegistration	Enable user self registration.	Online
configureUserSelfRegistrationAttr	Sets which attributes from the assertion should be used as email, first name, last name or username during self registration.	Online
createAuthnSchemeAndModule	Create an authentication scheme and module for an IdP partner.	Online
createIdPPartnerAttributeProfile	Create an IdP partner attribute profile for a federation partner.	Online
createSPPartnerAttributeProfile	Create an SP partner attribute profile for a federation partner.	Online
deleteAuthnSchemeAndModule	Delete an authentication scheme and module for an IdP partner.	Online
deleteFederationPartner	Delete a specific federation partner.	Online
deleteFederationPartnerEncryptionCert	Delete the encryption certificate of a federation partner.	Online
deleteFederationPartnerSigningCert	Delete the signing certificate of a federation partner.	Online
deleteIdPPartnerAttributeProfile	Delete the attribute profile of an IdP federation partner.	Online

Table 5–1 (Cont.) WLST Commands for Identity Federation

Use this command...	To...	Use with WLST...
deleteSPPartnerAttributeProfile	Delete the attribute profile of an SP federation partner.	Online
deleteIdPPartnerAttributeProfileEntry	Delete an entry from the attribute profile of a federation partner.	Online
deleteSPPartnerAttributeProfileEntry	Delete an entry from the attribute profile of a federation partner.	Online
deletePartnerProperty	Delete a partner-specific property that was added to the partner's configuration.	Online
displayIdPPartnerAttributeProfile	Display an IdP federation partner's attribute profile.	Online
displaySPPartnerAttributeProfile	Display an SP federation partner's attribute profile.	Online
getAllFederationIdentityProviders	List all IdP federation partners.	Online
getFederationPartnerEncryptionCert	Retrieve the encryption certificate for a federation partner.	Online
getFederationPartnerSigningCert	Retrieve the signing certificate for a federation partner	Online
getIdPPartnerBasicAuthCredentialUsername	Retrieve the HTTP basic authentication username for a federation partner.	Online
getPartnerProperty	Retrieve a property for a federation partner.	Online
getStringProperty	Retrieve a string property from a federation partner profile.	Online
isFederationPartnerPresent	Check whether a partner is configured.	Online
listIdPPartnerAttributeProfileIDs	List an IdP partner's attribute profiles.	Online
listSPPartnerAttributeProfileIDs	List an SP partner's attribute profiles.	Online
putStringProperty	Sets an OpenID partner as the default Federation IdP.	Online
setDefaultSSOIdPPartner	Set an IdP partner as the default identity provider for a federation single sign-on.	Online
setFederationPartnerEncryptionCert	Set the encryption certificate for a federation partner.	Online
setFederationPartnerSigningCert	Set the signing certificate for a federation partner.	Online
setIdPPartnerAttributeProfile	Set the attribute profile to use during federated single sign-on with an IdP partner.	Online
setIdPDefaultScheme	Sets the default OAM Authentication Scheme.	Online
setSPPartnerAttributeProfile	Set the attribute profile to use during federated single sign-on with an SP partner.	Online

Table 5–1 (Cont.) WLST Commands for Identity Federation

Use this command...	To...	Use with WLST...
setIdPPartnerAttributeProfileEntry	Set an entry in an IdP federation partner's profile.	Online
setSPPartnerAttributeProfileEntry	Set an entry in an SP federation partner's profile.	Online
setIdPPartnerBasicAuthCredential	Update a federation partner's HTTP basic auth credential.	Online
setIdPPartnerMappingAttribute	Set the attribute used for assertion mapping for a federation partner.	Online
setIdPPartnerMappingAttributeQuery	Set the attribute query used for assertion mapping for a federation partner.	Online
setIdPPartnerMappingNameID	Set the assertion mapping nameID value for an IdP federation partner	Online
setPartnerAlias	Update a federation partner's alias name.	Online
setPartnerIDStoreAndBaseDN	Set a federation partner's identity store and base DN.	Online
setSPPartnerAlternateScheme	Configure an alternate Authentication Scheme.	Online
setSPPartnerDefaultScheme	Configure a default Authentication Scheme.	Online
setSPPartnerProfileDefaultScheme	Configure the profile with a default Authentication Scheme.	Online
setSPPartnerProfileAlternateScheme	Configure the profile for an alternate Authentication Scheme.	Online
updatePartnerMetadata	Update a federation partner's metadata.	Online
updatePartnerProperty	Update a property for a federation partner.	Online

5.2 Advanced Identity Federation Commands

The Advanced Identity Federation WLST commands do not have applicable administrative fields for configuration in the Oracle Access Management Console. Administration for Authentication mappings and partner profiles are available using WLST commands only. Table 5–2 lists the Advanced Identity Federation commands documented in this section. The commands are organized as follows.

- Federation Service and Datastore
- Federation Access Configuration
- Attribute Sharing Configuration
- Authentication Method Mapping Management - All Authentication Method/Scheme/Level mappings are configured using WLST at the partner level or, if not defined at the partner level, at the partner profile level.
- Partner Profile Management - All Partner Profile management is done with WLST.
- Using WLST with SAML 1.1

Note: The Advanced Identity Federation command definitions begin with "configureFederationService" on page 5-68.

Table 5–2 Advanced Identity Federation WLST Commands

Use this command...	To...	Use with WLST...
Federation Service and Datastore		
configureFederationService	Enable or disable Federation Service features.	
setFederationStore	Enables and configures the federation store.	
Federation Access Configuration		
configureIdPAuthnRequest	Configure an IdP partner or IdP partner profile for Force Authentication and/or IsPassive.	
configureFedSSOAuthz	Enables or disables Authorization for Federation SSO.	
configureFedDigitalSignature	Configure the Hashing algorithm used in digital signatures.	
configureFedSignEncKey	Configure the signing and/or encryption key alias to be used for digital signature and encryption operations.	
Attribute Sharing Configuration		
configureAttributeSharingSPPartnerNameIDMapping	Configures the NameID to user store attribute mapping to be used during Attribute Sharing.	
configureAttributeSharingIdPPartner	Configures the default attribute sharing nameid and nameid format for the IdP Partner.	
configureAttributeSharingUserDNToIdPPartnerMapping	Configures Attribute Sharing DN to IdP Mappings.	
configureAttributeSharing	Configures the Attribute Sharing feature by setting a default attribute authority.	
removeAttributeSharingFromAuthnModule	Removes the Attribute Sharing plug-in from the Authentication Module.	
configureAttributeSharingPlugin	Lists the Federated Authentication Method mappings for a specific Partner Profile.	
insertAttributeSharingInToAuthnModule	Inserts the attribute sharing step into the Authentication Module flow.	
Authentication Method Mapping Management		
setSPPartnerAlternateScheme	Provides a way to authenticate clients with an alternate Authentication Scheme (Partner).	
setSPPartnerDefaultScheme	Defines the default Authentication Scheme for the SP partner.	

Table 5–2 (Cont.) Advanced Identity Federation WLST Commands

Use this command...	To...	Use with WLST...
setSPPartnerProfileAlternateScheme	Provides a way to authenticate clients with an alternate Authentication Scheme (Partner Profile).	
setSPPartnerProfileDefaultScheme	Sets the default OAM Authentication Scheme to be used to challenge a user for a specific SP Partner Profile.	
addSPPartnerAuthnMethod	Defines a mapping between a Federated Authentication Method and an Access Manager Authentication Scheme for a specific SP Partner.	
addSPPartnerProfileAuthnMethod	Defines a mapping between a Federated Authentication Method to an Access Manager Authentication Scheme for a specific SP Partner Profile.	
addIdPPartnerAuthnMethod	Sets the Authentication Level to use when creating a session for a Federated Authentication Method for a specific IdP Partner.	
addIdPPartnerProfileAuthnMethod	Sets the Authentication Level to use when creating a session for a Federated Authentication Method for a specific IdP Partner Profile.	
listPartnerAuthnMethods	Lists the Federated Authentication Method mappings for a specific Partner.	
listPartnerProfileAuthnMethods	Lists the Federated Authentication Method mappings for a specific Partner Profile.	
removePartnerAuthnMethod	Removes the mapping between a Federated Authentication Method and Access Manager Authentication Scheme for a specific Partner.	
removePartnerProfileAuthnMethod	Removes the mapping between a Federated Authentication Method and Access Manager Authentication Scheme for a specific Partner.	
setIdPPartnerRequestAuthnMethod	Sets the Federated Authentication Method that will be requested during Federation SSO for a specific IdP Partner.	
setIdPPartnerProfileRequestAuthnMethod	Sets the Federated Authentication Method that will be requested during Federation SSO for a specific IdP Partner Profile.	
useProxiedFedAuthnMethod	Configure the Identity Provider to use the proxied Federation Authentication Method when performing Federation SSO.	
Partner Profile Management		
createFedPartnerProfileFrom	Creates a Federation Partner Profile based on the specified existing one.	

Table 5–2 (Cont.) Advanced Identity Federation WLST Commands

Use this command...	To...	Use with WLST...
deleteFedPartnerProfile	Deletes the specified Federation Partner Profile.	
displayFedPartnerProfile	Displays the properties defined in the specified Federation Partner Profile.	
listFedPartnerProfiles	Lists all of the existing Federation Partner Profiles.	
listFedPartnersForProfile	Lists the partners bound to the specified Federation Partner Profile.	
getFedPartnerProfile	Gets the ID of the Partner Profile bound to the specified partner.	
setFedPartnerProfile	Sets the Federation Partner Profile ID for the specified partner.	
Using WLST with SAML 1.1		
When an IDP partner is configured for SAML 1.1, the following URL is used by the SP to start the SSO process. `http://idphost:idpport/ssourl?TARGET=targeturl&providerid=http://spproviderid` By using these WLST commands, the URL can be populated with the applicable information.		
idpinitiatedssoprovideridparam	Value is used by the peer provider to identify the provider ID of the SP.	
idpinitiatedssotargetparam	Sets the target URL for the specified SP partner.	
The following SAML 1.1 configuration parameters are not exposed through the Oracle Access Management Console. The values of these parameters can be modified using WLST.		
"deletePartnerProperty" on page 5-36	Delete a partner property.	
"getPartnerProperty" on page 5-44	Retrieve a partner property.	
"updatePartnerProperty" on page 5-66	Update a partner property.	
Subject Confirmation Check		
subjectconfirmationcheck	Enables or Disables the subject confirmation data check in SAML assertion.	

addWSFed11IdPFederationPartner

Creates a WS-Federation 1.1 IdP partner.

Description

Creates an IdP partner under the WS-Federation 1.1 protocol. The NameID will be mapped to the LDAP user mail attribute.

Syntax

```
addWSFed11IdPFederationPartner(partnerName,ssoURL, providerID, description)
```

Argument	Definition
partnerName	The name of the partner to be created.
ssoURL	The Identity Realm Secure Token URL where users will be redirected at the IdP for WS-Federation 1.1 operations.
providerID	Provider ID/Issuer used in the SAML Assertion.
description	The description of the partner. Optional.

Example

```
addWSFed11IdPFederationPartner("testpartner1", "http://idp.com/wsfed11",
 "http://idp.com", description="WS-Fed IdP1")
```

addWSFed11SPFederationPartner

Creates a WS-Federation 1.1 SP partner.

Description

Creates an SP partner under the WS-Federation 1.1 protocol.

Syntax

```
addWSFed11SPFederationPartner(partnerName, realm, ssoURL, samlVersion,
msftADFSCompatible, description)
```

Argument	Definition
partnerName	The name of the partner to be created.
realm	The realm identifier for this SP partner. It will be used in the WS-Federation 1.1 protocol exchange.
ssoURL	The Identity Realm Secure Token URL where users will be redirected at the SP for WS-Federation 1.1 operations.
samlVersion	The optional SAML version indicating what kind of Assertion to issue. Takes a value of saml11 (default) or saml20.
msftADFSCompatible	An optional boolean indicating if the issued SSO Response should be in the Microsoft ADFS compatible format WS-Trust 1.2 or WS-Trust 1.3.
description	The description of the partner. Optional.

Example

```
addWSFed11SPFederationPartner("testpartner1", "http://sp.com",
 "http://sp.com/wsfed11", description="Test SP1")
```

addOpenID20IdPFederationPartner

Creates an OpenID 2.0 IdP partner.

Description

Creates an IdP partner under the OpenID 2.0 protocol.

Syntax

```
addOpenID20IdPFederationPartner(partnerName, idpSSOURL, discoveryURL, description)
```

Argument	Definition
partnerName	The name of the partner to be created.
idpSSOURL	The initiate SSO URL of the IdP. Can be set to "" if the discovery URL is specified and intended to be used.
discoveryURL	The OpenID discovery URL of the IdP.
description	The description of the partner. Optional.

Example

```
addOpenID20IdPFederationPartner("testpartner1", "",
 "http://host:port/discoveryurl", description="Test IdP1")
```

addOpenID20SPFederationPartner

Creates an OpenID 2.0 SP partner.

Description

Creates an SP partner under the OpenID 2.0 protocol.

Syntax

```
addOpenID20SPFederationPartner(partnerName, realm, ssoURL, description)
```

Argument	Definition
partnerName	The name of the partner to be created.
realm	The realm for the SP (RP).
ssoURL	The endpoint URL of the SP (RP).
description	The description of the partner. Optional.

Example

```
addOpenID20SPFederationPartner(partnerName="partnerID",
 realm="http://realm.domain.com", ssoURL="http://host:port/endpoint",
 description="some description")
```

addOpenID20GoogleIdPFederationPartner

Creates an IdP partner with the name `google`.

Description

Creates an IdP partner with the name `google` using a discovery URL `https://www.google.com/accounts/o8/id`.

Syntax

```
addOpenID20GoogleIdPFederationPartner()
```

Example

```
addOpenID20GoogleIdPFederationPartner()
```

addOpenID20YahooIdPFederationPartner

Creates an IdP partner with the name `yahoo`.

Description

create an IdP partner with the name `yahoo` using a discovery URL `https://open.login.yahooapis.com/openid20/user_profile/xrds`.

Syntax

```
addOpenID20YahooIdPFederationPartner()
```

Example

```
addOpenID20YahooIdPFederationPartner()
```

addSAML11IdPFederationPartner

Creates a SAML 1.1 IdP federation partner.

Description

Creates a SAML 1.1 IdP federation partner.

Syntax

```
addSAML11IdPFederationPartner(partnerName,providerID, ssoURL,
soapURL, succinctID, description)
```

Argument	Definition
partnerName	The name of the partner to be created.
providerID	The providerID of the partner.
ssoURL	The initiate SSO URL of the IdP.
soapURL	The artifact resolution SOAP endpoint URL of the IdP.
succinctID	The succinctID of the provider.
description	The description of the partner. Optional.

Example

```
addSAML11IdPFederationPartner(partnerName="partnerID",
providerID="providerA", ssoURL="http://host:port/saml11sso",
soapURL="http://host:port/soapurl", succinctID="1234",
description="somedescription")
```

addSAML11SPFederationPartner

Creates a SAML 1.1 SP federation partner.

Description

Creates a SAML 1.1 SP federation partner.

Syntax

```
addSAML11SPFederationPartner(partnerName,providerID, ssoURL, description)
```

Argument	Definition
partnerName	The name of the partner to be created.
providerID	The providerID of the partner.
ssoURL	The initiate SSO URL of the IdP.
description	The description of the partner. Optional.

Example

```
addSAML11SPFederationPartner(partnerName="partnerID", providerID="providerA",
ssoURL="http://host:port/saml11sso", description="somedescription")
```

addSAML20IdPFederationPartner

Creates a SAML 2.0 IdP Federation partner.

Description

Creates a federation partner as an identity provider for Access Manager under the SAML 2.0 protocol, and loads the partner metadata from a file.

Syntax

```
addSAML20IdPFederationPartner(partnerName, metadataFile, description)
```

Argument	Definition
partnerName	The name of the partner to be created.
metadataFile	The location of the metadata file (full path).
description	The description of the partner. Optional.

Example

```
addSAML20IdPFederationPartner(partnerName="partnerID",
metadataFile="location_metadata_file", description="somedescription")
```

addSAML20SPFederationPartner

Creates a SAML 2.0 SP Federation partner.

Description

Creates a federation partner as a service provider for Access Manager under the SAML 2.0 protocol, and loads the partner metadata from a file.

Syntax

```
addSAML20SPFederationPartner(partnerName, metadataFile, description)
```

Argument	Definition
partnerName	The name of the partner to be created.
metadataFile	The location of the metadata file (full path).
description	The description of the partner. Optional.

Example

```
addSAML20SPFederationPartner(partnerName="partnerID",
metadataFile="location_metadata_file", description="somedescription")
```

addSAML20IdPFederationPartnerWithoutMetadata

Creates a SAML20 IdP federation partner without SAML 2.0 metadata.

Description

Creates a SAML20 IdP federation partner without loading SAML 2.0 metadata.

Syntax

```
addSAML20IdPFederationPartnerWithoutMetadata(partnerName,
providerID, ssoURL, soapURL, succinctID, description)
```

Argument	Definition
partnerName	The name of the federation partner to be created.
providerID	The providerID of the partner.
ssoURL	The initiate SSO URL of the IdP.
soapURL	The artifact resolution SOAP endpoint URL of the IdP.
succinctID	The succinctID of the provider.
description	The description of the partner. Optional.

Example

```
addSAML20IdPFederationPartnerWithoutMetadata(partnerName="partnerName",
providerID="http://host:port", ssoURL="http://host:port/saml/sso",
soapURL="http://host:port/saml/soap",description="some description")
```

addSAML20SPFederationPartnerWithoutMetadata

Creates a SAML20 SP federation partner without SAML 2.0 metadata.

Description

Creates a SAML20 SP federation partner without loading SAML 2.0 metadata.

Syntax

```
addSAML20SPFederationPartnerWithoutMetadata(partnerName,
providerID, ssoURL, description)
```

Argument	Definition
partnerName	The name of the federation partner to be created.
providerID	The providerID of the partner.
ssoURL	The initiate SSO URL of the IdP.
description	The description of the partner. Optional.

Example

```
addSAML20SPFederationPartnerWithoutMetadata(partnerName="partnerName",
providerID="http://host:port", ssoURL="http://host:port/saml/sso",
description="somedescription")
```

configureIdPPartnerAttributeProfile

Configures an IdP partner attribute profile to process incoming attributes.

Description

Configures an IdP partner attribute profile to process or ignore incoming attributes not defined in the profile.

Syntax

```
configureIdPPartnerAttributeProfile(attrProfileID, ignoreUnmappedAttributes)
```

Argument	Definition
attrProfileID	The identifier referencing the IdP partner attribute profile to configure.
ignoreUnmappedAttributes	Determines whether incoming attributes that are not defined in the profile should be ignored. Valid values are true (ignore) or (the default) false (process).

Example

```
configureIdPPartnerAttributeProfile(attrProfileID="idp-attribute-profile",
ignoreUnmappedAttributes="false")
```

configureSAML20Logout

Configures global federation logout for a SAML 2.0 partner.

Description

Configures global federation logout for a SAML 2.0 federation partner.

Syntax

```
configureSAML20Logout(partnerName, partnerType, enable,
saml20LogoutRequestURL, saml20LogoutResponseURL, soapURL)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
partnerType	Whether the partner is a service provider or identity provider. Valid values are sp, idp.
enable	Enable or disable global logout for that partner. Valid values true (enable), false (disable)
saml20LogoutRequestURL	The SAML 2.0 logout request service URL. Optional if the partner was created using metadata, or if logout is disabled.
saml20LogoutResponseURL	The SAML 2.0 logout response service URL. This is optional if the partner was created using metadata, or if logout is disabled.
soapURL	The SAML 2.0 SOAP Service URL. This is optional if the partner was created using metadata, if logout is disabled, or if SOAP logout is not supported.

Example

```
configureSAML20Logout(partnerName="partnerID", partnerType="sp", enable="true",
saml20LogoutRequestURL="http://host:port/saml/logoutrequest",
saml20LogoutResponseURL="http://host:port/saml/logoutresponse",
soapURL="http://host:port/saml/soap")
```

configureSAMLBinding

Specifies the binding for a SAML partner.

Description

Configures the preferred binding for a SAML Partner.

Syntax

```
configureSAMLBinding(partnerName, partnerType, binding,
ssoResponseBinding="httppost")
```

Argument	Definition
partnerName	The name of the partner to be configured.
partnerType	Indicates whether the partner is a service provider or an identity provider. Valid values are sp, idp.
binding	Specifies the binding to use for messages other than SSO responses (authentication requests, logout messages). Valid options are httppost for HTTP-POST binding and httpredirect for HTTP-Redirect binding.
ssoResponseBinding	This optional attribute defines the binding to use for an SSO response. Valid options are httppost for HTTP-POST binding (the default value), httpredirect for HTTP-Redirect binding or artifact for Artifact binding.

Example

```
configureSAMLBinding(partnerName="partnerID",
partnerType="sp", binding="httpredirect", ssoResponseBinding="httppost")
```

configureUserSelfRegistration

Enables the user self-registration module.

Description

Enables the user self-registration module.

Syntax

```
configureUserSelfRegistration(<enabled>, <registrationURL>,
 <regDataRetrievalAuthnEnabled>, <regDataRetrievalAuthnUsername>,
 <regDataRetrievalAuthnPassword>, <partnerName>)
```

Argument	Definition
enabled	Indicates if the user self-registration module is enabled. Takes a value of true or false.
registrationURL	The location to which the user will be redirected for self-registration. If partnerName is not specified, and if registrationURL is empty or missing, the current property will be unchanged. If partnerName is specified, and if registrationURL is empty or missing, this property will be removed from the partner's configuration.
regDataRetrievalAuthnEnabled	Indicates if authentication of the registration page is enabled when contacting the server to retrieve registration data.
regDataRetrievalAuthnUsername	Specifies the username the registration page will send to the server when retrieving the registration data from the server.
regDataRetrievalAuthnPassword	Specifies the password the registration page will send to the server when retrieving the registration data from the server.
partnerName	Indicates the IdP partner for which to enable user self-registration. If missing, the configuration operation will be global.

Example

```
configureUserSelfRegistration("true", regDataRetrievalAuthnEnabled="true",
 regDataRetrievalAuthnUsername="username",
 regDataRetrievalAuthnPassword="password")
```

configureUserSelfRegistrationAttr

Sets the attributes in an assertion that will be used as email, first name, last name and username.

Description

Sets the attributes in an assertion that will be used as email, first name, last name and username.

Syntax

```
configureUserSelfRegistration(<registrationAttrName>, <assertionAttrNames>,
 <partnerName>)
```

Argument	Definition
registrationAttrName	The self-registration page attribute to set. Can be one of the following values: email, firstname, lastname or username.
assertionAttrNames	The possible attributes from the assertion that can be used to populate the self-registration page field specified as the registrationAttrName.
partnerName	Indicates the IdP partner for which to configure user self-registration. If missing, the configuration operation will be global.

Example

```
configureUserSelfRegistrationAttr("email", "mail,fed.nameidvalue")
```

The second parameter means that *mail* or *fed.nameidvalue* from the assertion can be used to populate the email attribute in the user's self registration page.

createAuthnSchemeAndModule

Creates an authentication scheme that uses an OpenD IdP.

Description

Creates an authentication scheme that uses an OpenD IdP to protect resources in Access Manager.

Syntax

```
createAuthnSchemeAndModule(partnerName)
```

Argument	Definition
partnerName	The name of the partner for whom the scheme is to be created.

Example

```
createAuthnSchemeAndModule("testpartner")
```

createIdPPartnerAttributeProfile

Creates an IdP attribute profile.

Description

Creates an IdP partner attribute profile that will contain name mapping rules used to process attributes in incoming SAML Assertions

Syntax

```
createIdPPartnerAttributeProfile(attrProfileID)
```

Argument	Definition
attrProfileID	The identifier of the IdP attribute profile.

Example

```
createIdPPartnerAttributeProfile(attrProfileID="idp-attribute-profile")
```

createSPPartnerAttributeProfile

Creates an SP attribute profile.

Description

Creates an SP partner attribute profile that will contain name mapping rules used to process attributes in incoming SAML Assertions

Syntax

```
createSPPartnerAttributeProfile(attrProfileID)
```

Argument	Definition
attrProfileID	The identifier of the SP attribute profile.

Example

```
createSPPartnerAttributeProfile(attrProfileID="sp-attribute-profile")
```

deleteAuthnSchemeAndModule

Deletes an authentication scheme for an IdP.

Description

Deletes an authentication scheme for an IdP partner.

Syntax

```
deleteAuthnSchemeAndModule(partnerName)
```

Argument	Definition
partnerName	The name of the partner whose scheme is to be deleted.

Example

```
deleteAuthnSchemeAndModule("testpartner")
```

deleteFederationPartner

Deletes a federation partner.

Description

Deletes a federation partner from Access Manager.

Syntax

```
deleteFederationPartner(partnerName, partnerType)
```

Argument	Definition
partnerName	The ID of the partner to be deleted.
partnerType	Specifies whether the partner is a service provider or an identity provider. Valid values are sp, idp.

Example

```
deleteFederationPartner(partnerName="partnerID", partnerType="idp")
```

deleteFederationPartnerEncryptionCert

Deletes the encryption certificate of a partner.

Description

Deletes the encryption certificate of a federation partner.

Syntax

```
deleteFederationPartnerEncryptionCert(partnerName, partnerType)
```

Argument	Definition
partnerName	The ID of the partner whose encryption certificate is to be deleted.
partnerType	Specifies whether the partner is a service provider or an identity provider. Valid values are sp, idp.

Example

```
deleteFederationPartnerEncryptionCert(partnerName="customPartner",
partnerType="idp")
```

deleteFederationPartnerSigningCert

Deletes the signing certificate of a partner.

Description

Deletes the signing certificate of a federation partner.

Syntax

```
deleteFederationPartnerSigningCert(partnerName, partnerType)
```

Argument	Definition
partnerName	The ID of the partner whose signing certificate is to be deleted.
partnerType	Specifies whether the partner is a service provider or identity provider. Valid values are sp, idp.

Example

```
deleteFederationPartnerSigningCert(partnerName="customPartner",partnerType="idp")
```

deleteIdPPartnerAttributeProfile

Deletes an IdP partner attribute profile.

Description

Deletes an IdP partner attribute profile.

Syntax

```
deleteIdPPartnerAttributeProfile(attrProfileID)
```

Argument	Definition
attrProfileID	The identifier referencing the IdP partner attribute profile.

Example

```
deleteIdPPartnerAttributeProfile(attrProfileID="idp-attribute-profile")
```

deleteSPPartnerAttributeProfile

Deletes an SP partner attribute profile.

Description

Deletes an SP partner attribute profile.

Syntax

```
deleteSPPartnerAttributeProfile(attrProfileID)
```

Argument	Definition
attrProfileID	The identifier referencing the SP partner attribute profile.

Example

```
deleteSPPartnerAttributeProfile(attrProfileID="sp-attribute-profile")
```

deleteIdPPartnerAttributeProfileEntry

Deletes an IdP Partner Attribute Profile entry.

Description

Deletes an attribute from the attribute profile.

Syntax

```
deleteIdPPartnerAttributeProfileEntry(attrProfileID,
messageAttributeName)
```

Argument	Definition
attrProfileID	The identifier referencing the IdP partner attribute profile.
messageAttributeName	The name of the attribute to delete, as it appears in the outgoing message.

Example

```
deleteIdPPartnerAttributeProfileEntry(attrProfileID="idp-attribute-profile",
messageAttributeName="first_name")
```

deleteSPPartnerAttributeProfileEntry

Deletes an SP Partner Attribute Profile entry.

Description

Deletes an attribute from the attribute profile.

Syntax

```
deleteSPPartnerAttributeProfileEntry(attrProfileID,
 messageAttributeName)
```

Argument	Definition
attrProfileID	The identifier referencing the IdP partner attribute profile.
messageAttributeName	The name of the attribute to delete, as it appears in the outgoing message.

Example

```
deleteSPPartnerAttributeProfileEntry(attrProfileID="sp-attribute-profile",
 messageAttributeName="first_name")
```

deletePartnerProperty

Deletes a partner property.

See Advanced Identity Federation Commands for information regarding SAML 1.1.

Description

Deletes a partner-specific property. Use this command only for a property that was added to the partner's configuration.

Syntax

```
deletePartnerProperty(partnerName,partnerType,propName)
```

Argument	Definition
partnerName	The ID of the partner to be updated. By replacing the value of <partnerName> with the partner ID and including the `includecertinsignature` parameter, the certificate will be included with the signature. See Advanced Identity Federation Commands for information regarding SAML 1.1.
partnerType	Specifies whether the partner is a service provider or an identity provider. Valid values are sp, idp.
propName	The name of the configured property to be removed.

Example

```
deletePartnerProperty(partnerName="partner1025", partnerType="sp/idp",
propName="includecertinsignature")
```

displayIdPPartnerAttributeProfile

Displays a partner attribute profile.

Description

Display the content of an IdP Partner Attribute Profile.

Syntax

```
displayIdPPartnerAttributeProfile(attrProfileID)
```

Argument	Definition
attrProfileID	The identifier referencing the IdP partner attribute profile to be displayed.

Example

```
displayIdPPartnerAttributeProfile(attrProfileID="idp-attribute-profile")
```

displaySPPartnerAttributeProfile

Displays an SP partner attribute profile.

Description

Display the content of an SP Partner Attribute Profile.

Syntax

```
displaySPPartnerAttributeProfile(attrProfileID)
```

Argument	Definition
attrProfileID	The identifier referencing the SP partner attribute profile to be displayed.

Example

```
displaySPPartnerAttributeProfile(attrProfileID="sp-attribute-profile")
```

getAllFederationIdentityProviders

Lists all federation identity providers.

Description

Displays a list of all federation identity providers for Access Manager.

Syntax

```
getAllFederationIdentityProviders()
```

Example

```
getAllFederationIdentityProviders()
```

getAllFederationServiceProviders

Lists all federation service providers.

Description

Displays a list of all federation service providers for Access Manager.

Syntax

```
getAllFederationServiceProviders()
```

Example

```
getAllFederationServiceProviders()
```

getFederationPartnerEncryptionCert

Retrieves the encryption certificate for a partner.

Description

Retrieves the encryption certificate for a federation partner.

Syntax

Argument	Definition
partnerName	The ID of the partner for which the encryption certificate will be retrieved.
partnerType	Specifies whether the partner is a service provider or an identity provider. Valid values are sp, idp.

Example

```
getFederationPartnerEncryptionCert(partnerName="customPartner",partnerType="idp")
```

getFederationPartnerSigningCert

Retrieves the signing certificate for a partner.

Description

Retrieves the signing certificate for a federation partner.

Syntax

Argument	Definition
partnerName	The ID of the partner for which the signing certificate will be retrieved.
partnerType	Specifies whether the partner is a service provider or an identity provider. Valid values are sp, idp.

Example

```
getFederationPartnerSigningCert(partnerName="partnerID1", partnerType="idp")
```

getIdPPartnerBasicAuthCredentialUsername

Gets a partner's basic authentication username.

Description

Retrieves the HTTP basic authentication username for a federation partner.

Syntax

```
getIdPPartnerBasicAuthCredentialUsername(partnerName)
```

Argument	Definition
partnerName	The ID of the partner for which the username will be retrieved and displayed.

Example

```
getIdPPartnerBasicAuthCredentialUsername(partnerName="partnerID5")
```

getPartnerProperty

Retrieves a partner property.

Description

Retrieves a property for a federation partner.

Syntax

```
getPartnerProperty(partnerName, partnerType, propName)
```

Argument	Definition
partnerName	The ID of the partner for which the proeprty will be retrieved. By replacing the value of <partnerName> with the partner ID and including the `includecertinsignature` parameter, the certificate will be included with the signature. See Advanced Identity Federation Commands for information regarding SAML 1.1.
partnerType	Specifies whether the partner is a service provider or an identity provider. Valid values are sp, idp.
propName	The name of the property to configure.

Example

```
getPartnerProperty(partnerName="partnerID4", partnerType="sp",
 propName="providertrusted")
```

getStringProperty

Retrieves a string property.

Description

Retrieves a string property for a federation partner profile.

If a Partner does not have an Attribute Profile assigned to it, the default Attribute Profile (based on whether the partner is an IdP or SP) will be used. The `defaultattributeprofileidp` and `defaultattributeprofilesp` properties in the `fedserverconfig` file reference the default profiles.

Syntax

```
getStringProperty("/fedserverconfig/<propertyName>")
```

Argument	Definition
propertyName	The name of the property to be retrieved. Default Partner Profiles are available after installation and the following properties reference them. Default property values can be retrieved by replacing *propertyName* with one of the following: ■ defaultpartnerprofileidpsaml20: default Partner Profile for SAML 2.0 IdP Partners ■ defaultpartnerprofilespsaml20: default Partner Profile for SAML 2.0 SP Partners ■ defaultpartnerprofileidpsaml11: default Partner Profile for SAML 1.1 IdP Partners ■ defaultpartnerprofilespsaml11: default Partner Profile for SAML 1.1 SP Partners ■ defaultpartnerprofileidpopenid20: default Partner Profile for OpenID 2.0 IdP Partners ■ defaultpartnerprofilespopenid20: default Partner Profile for OpenID 2.0 SP Partners ■ If : "defaultattributeprofileidp: default Attribute Profile for IdP Partners "defaultattributeprofilesp: default Attribute Profile SP Partners

Example

```
getStringProperty("/fedserverconfig/defaultpartnerprofileidpopenid20")
```

isFederationPartnerPresent

Checks whether a partner is configured.

Description

Checks whether the specified federation partner is defined in Access Manager.

Syntax

```
isFederationPartnerPresent(partnerName, partnerType)
```

Argument	Definition
partnerName	The partner ID.
partnerType	Specifies whether the partner is a service provider or an identity provider. Valid values are sp, idp.

Example

```
isFederationPartnerPresent(partnerABC, SP)
```

listIdPPartnerAttributeProfileIDs

Lists the IdP partner attribute profiles.

Description

List the identifiers of the existing IdP Partner Attribute Profiles.

Syntax

```
listIdPPartnerAttributeProfileIDs()
```

Example

```
listIdPPartnerAttributeProfileIDs()
```

listSPPartnerAttributeProfileIDs

Lists the SP partner attribute profiles.

Description

List the identifiers of the existing SP Partner Attribute Profiles.

Syntax

```
listSPPartnerAttributeProfileIDs()
```

Example

```
listSPPartnerAttributeProfileIDs()
```

putStringProperty

Puts a string value under a designated path in the OSTS configuration.

Description

Puts a string value under a designated path in the OSTS configuration.

Syntax

```
putStringProperty(path="/validationtemplates/username-wss-validation-template/Stri
ngNAME",value="TestString")
```

Argument	Definition
path	Path inside the configuration where the String property will be put.
value	The string.

Example

```
putStringProperty("/spglobal/defaultssoidp", "testpartner")
```

setDefaultSSOIdPPartner

Sets the IdP partner to serve as the default IdP for federated single sign-on (SSO).

Description

If not set by the federation authentication plugin at run time, sets the IdP partner to serve as the default IdP during federated SSO.

Syntax

```
setDefaultSSOIdPPartner(partnerName)
```

Argument	Definition
partnerName	ID of the partner which will serve as the default IdP for federated SSO.

Example

```
setDefaultSSOIdPPartner(partnerName="partner25")
```

setFederationPartnerEncryptionCert

Sets the encryption certificate for a partner.

Description

Sets the encryption certificate for a federation partner.

Syntax

```
setFederationPartnerEncryptionCert(partnerName,partnerType,certFile)
```

Argument	Definition
partnerName	The ID of the partner to be updated
partnerType	The partner type. Valid values are idp, sp.
certFile	The full path and name of file that stores the encryption certificate. Certificates can be in either PEM or DER format.

Example

```
setFederationPartnerEncryptionCert
(partnerName="customPartner",partnerType="idp",
certFile="/temp/encryption_cert")
```

setFederationPartnerSigningCert

Sets the signing certificate for a partner.

Description

Sets the signing certificate for a federation partner.

Syntax

```
setFederationPartnerSigningCert(partnerName,partnerType,certFile)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
partnerType	The partner type. Valid values are idp, sp.
certFile	Specifies the full path and name of file that stores the signing certificate. Certificates can be in either PEM or DER format.

Example

```
setFederationPartnerSigningCert
(partnerName="customPartner", partnerType="idp",
certFile="/temp/signing_cert")
```

setIdPPartnerAttributeProfile

Sets a partner attribute profile.

Description

Sets the IdP partner attribute profile to use when performing a federation single sign-on with an IdP partner.

Syntax

```
setIdPPartnerAttributeProfile(partnerName, attrProfileID)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
attrProfileID	The IdP partner attribute profile ID to be set.

Example

```
setIdPPartnerAttributeProfile(partnerName="partnerID5",
attrProfileID="idp-attribute-profile")
```

setIdPDefaultScheme

Sets the default OAM Authentication Scheme to be used to challenge a user.

Description

Sets the default OAM Authentication Scheme that will be used to challenge a user.

Syntax

```
setIdPDefaultScheme(authnScheme, appDomain, hostID,
 authzPolicy="ProtectedResourcePolicy")
```

Argument	Definition
authnScheme	The OAM Authentication Scheme.
appDomain	Optional. The application domain in which the underlying policy components will be created.
hostID	Optional. The HostID to be used when creating the underlying resource policy object.
authzPolicy	Optional. The name of the Authorization Policy to be used to protect underlying resource policy object being created.

Example

```
setIdPDefaultScheme('LDAPScheme')
```

Prepend the command with "fed." if running on the WebSphere platform.

setSPPartnerAttributeProfile

Sets an SP partner attribute profile to an SP partner.

Description

Sets the SP partner attribute profile to use with an SP partner.

Syntax

```
setSPPartnerAttributeProfile(partnerName, attrProfileID)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
attrProfileID	The ID of the SP partner attribute profile to be set.

Example

```
setSPPartnerAttributeProfile(partnerName="partnerID5",
attrProfileID="sp-attribute-profile")
```

setIdPPartnerAttributeProfileEntry

Sets the IdP federation partner profile.

Description

Update an entry in the IdP Partner Attribute Profile.

Syntax

```
setIdPPartnerAttributeProfileEntry(attrProfileID, messageAttributeName,
oamSessionAttributeName, requestFromIdP)
```

Argument	Definition
attrProfileID	The IdP partner attribute profile.
messageAttributeName	The name of the message attribute.
oamSessionAttributeName	The name of the attribute as it will appear in the Access Manager session.
requestFromIdP	Determines whether this attribute should be requested from the IdP partner. Valid values are true, false.

Example

```
setIdPPartnerAttributeProfileEntry(attrProfileID="idp-attribute-profile",
messageAttributeName="first_name",
oamSessionAttributeName="first_name", requestFromIdP="true")
```

setSPPartnerAttributeProfileEntry

Sets the SP federation partner profile.

Description

Sets an entry in the SP Partner Attribute Profile.

Syntax

```
setSPPartnerAttributeProfileEntry(attrProfileID, messageAttributeName,
value, alwaysSend)
```

Argument	Definition
attrProfileID	The identifier referencing the SP Partner Attribute Profile in which the entry will be set.
messageAttributeName	The name of the attribute as it will appear in the outgoing message.
value	Value of the attribute element. It can be a static string, user attribute, session attribute or a combination of those types.
alwaysSend	Signifies whether or not this attribute should always be sent to the SP Partner. Valid values are true, false. If false it will only be sent if the SP Partner requests it (OpenID supports this).

Example

```
setSPPartnerAttributeProfileEntry(attrProfileID="sp-attribute-profile",
 messageAttributeName="first_name", value="$user.attr.givenname",
 alwaysSend="true")
```

setIdPPartnerBasicAuthCredential

Sets a partner's basic authentication credentials.

Description

Sets or updates a federation partner's HTTP basic authentication credentials.

Syntax

```
setIdPPartnerBasicAuthCredential(partnerName,username,password)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
username	The user ID of the user.
password	The password corresponding to the username.

Example

```
setIdPPartnerBasicAuthCredential(partnerName="partnerID4", username="user1")
```

setIdPPartnerMappingAttribute

Sets a partner's assertion mapping attribute.

Description

Specify that an attribute from the OpenID assertion received from the IdP be mapped to a given data store attribute in order to identify the user.

Syntax

```
setIdPPartnerMappingAttribute(partnerName,assertionAttr,userstoreAttr)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
assertionAttr	The attribute name in the assertion used to map the user to the identity store.
userstoreAttr	The name of the attribute in the identity store to which to map the assertion attribute value.

Example

```
setIdPPartnerMappingAttribute(partnerName="partnerID",
assertionAttr="email", userstoreAttr="mail")
```

setIdPPartnerMappingAttributeQuery

Updates a partner for assertion mapping of user with attribute query.

Description

Sets or updates a partner to specify the attribute query to map an assertion to the user store.

Syntax

```
setIdPPartnerMappingAttributeQuery(partnerName,attrQuery)
```

Argument	Definition
partnerName	The ID of the partner to be updated
attrQuery	The attribute query to be used. The LDAP query can contain placeholders referencing the attributes in the SAML Assertion, as well as the NameID. An attribute from the SAML Assertion will be referenced by its name and surrounded by the % character; for example, if the attribute name is Userlastname, the attribute will be referenced as %Userlastname%. The NameID Value is referenced as %fed.nameidvalue%.

Example

```
setIdPPartnerMappingAttributeQuery(partnerName="partnerID",
attrQuery="(&(sn=%Userlastname%)(givenname=%Userfirstname%))")
```

setIdPPartnerMappingNameID

Sets a partner's mapping nameID.

Description

Sets the assertion mapping nameID value for an IdP federation partner.

Syntax

```
setIdPPartnerMappingNameID(partnerName,userstoreAttr)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
userstoreAttr	The attribute name in the identity store to which the assertion nameID is to be mapped.

Example

```
setIdPPartnerMappingNameID
(partnerName="partnerID", userstoreAttr="ldapattr")
```

setPartnerAlias

Sets a partner's alias.

Description

Sets or updates a federation partner's alias.

Syntax

```
setPartnerAlias(partnerName,partnerType,partnerAlias)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
partnerType	Specifies the partner type. Valid values are sp or idp.
partnerAlias	The partner's alias.

Example

```
setPartnerAlias(partnerName="partnerID",
partnerType="sp", partnerAlias="tenant1")
```

setPartnerIDStoreAndBaseDN

Sets a partner's identity store and base DN.

Description

Sets or updates the identity store and base DN of a federation partner.

Syntax

```
setPartnerIDStoreAndBaseDN(partnerName,partnerType,storeName,searchBaseDN)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
partnerType	The partner type. Valid values are sp or idp.
storeName	The name of the identity store.If left blank, the Default OAM Identity Store will be used. (Optional)
searchBaseDN	The search base DN for the LDAP. If left blank, the Search Base DN configured in the Identity Store will be used. (Optional)

Example

```
setPartnerIDStoreAndBaseDN(partnerName="partnerID",
 partnerType="sp/idp", storeName="testldap",
 searchBaseDN="dc=company,dc=com")
```

setSPSAMLPartnerNameID

Updates a partner by setting the NameID during assertion issuance.

Description

Sets the NameID for a SAML partner.

Syntax

```
setSPSAMLPartnerNameID(<partnerName>, <nameIDFormat>, <nameIDValue>)
```

Argument	Definition
partnerName	The name of the partner to be configured.
nameIDFormat	The NameID format to be used. Possible values include: ■ orafed-emailaddress for urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress ■ orafed-x509 for urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName ■ orafed-kerberos for urn:oasis:names:tc:SAML:2.0:nameid-format:Kerberos ■ orafed-transient for urn:oasis:names:tc:SAML:2.0:nameid-format:transient ■ orafed-windowsnamequalifier for urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName ■ orafed-persistent for urn:oasis:names:tc:SAML:2.0:nameid-format:persistent ■ orafed-unspecified for urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified ■ orafed-none for no NameID ■ If the format is set to any other value, the Assertion will be populated with that value.
nameIDValue	Value of the NameID element. It can be a static string, user attribute, session attribute or a combination of those types.

Example

```
setSPSAMLPartnerNameID(partnerName="partnerID", nameIDFormat="emailAddress",
 nameIDValue="$user.attr.mail")
```

updatePartnerMetadata

Updates partner metadata.

Description

Updates the metadata for a federation partner.

Syntax

```
updatePartnerMetadata(partnerName,partnerType,metadataFile)
```

Argument	Definition
partnerName	The ID of the partner to be updated
partnerType	Specifies the partner type. Valid values are sp or idp.
metadataFile	The location of the metadata file. Specify the complete path and name.

Example

```
updatePartnerMetadata(partnerName="partnerID",
partnerType="sp", metadataFile="/common/idm/abc_metadata_file")
```

updatePartnerProperty

Updates a partner property.

See Advanced Identity Federation Commands for information regarding SAML 1.1.

Description

Configures or updates the specified property for a federation partner.

Syntax

```
updatePartnerProperty(partnerName,partnerType,propName,propValue,type)
```

Argument	Definition
partnerName	The ID of the partner to be updated. By replacing the value of <partnerName> with the partner ID and including the `includecertinsignature` parameter, the certificate will be included with the signature. See Advanced Identity Federation Commands for information regarding SAML 1.1.
partnerType	Specifies the partner type. Valid values are sp or idp.
propName	The name of the property to configure.
propValue	The property value to be set.
type	The data type of the property. Valid values are string, long, or boolean.

Example

```
updatePartnerProperty(partnerName="partnerID", partnerType="idp",
propName="providertrusted",
propValue="true",type="boolean")
```

subjectconfirmationcheck

Enable or disable the Subject Confirmation Data check.

Description

Enable or disable the Subject Confirmation Data check in SAML assertion.

Syntax

```
updatePartnerProperty(partnerName,partnerType,propName,propValue,type)
```

Argument	Definition
partnerName	The ID of the partner to be updated.
partnerType	Specifies the partner type. Valid values are sp or idp.
propName	Set the property name as 'subjectconfirmationcheck'.
propValue	Specify the property value. Valid values are true or false.
type	Data type of the property. It can only be boolean.

Example

```
updatePartnerProperty(partnerName="testIDP", partnerType="IDP",
propName="subjectconfirmationcheck",
propValue="true",type="boolean")
```

configureFederationService

Enable or disable the Federation Service AttributeRequester or AttributeResponder.

Description

Enable or disable Federation Service features.

Syntax

```
configureFederationService(<serviceType>,<enabled>)
```

Argument	Definition
serviceType	Takes as a value IDP, SP, AttributeResponder or AttributeRequester.
enabled	Takes as a value either true or false.

Example

```
configureFederationService("idp", "true")

configureFederationService("AttributeResponder", "true")
```

setFederationStore

Enables and configures for the use of the federation store.

Description

This will set the jndiname of the datastore to be used to store federation records and will set the store as a RDBMS.

Syntax

```
setFederationStore (<enable>, <jndiname>)
```

Argument	Definition
enable	Enable or disable the Federation data store.
jndiname	Indicates the JNDI name of the datastore.

Example

```
setFederationStore(enable="true", jndiname="jdbc/oamds")
```

configureIdPAuthnRequest

Configure an IdP partner or an IdP partner profile for Force Authentication and/or IsPassive.

Description

Configure an IdP partner or IdP partner profile for Force Authentication and/or IsPassive.

Syntax

```
configureIdPAuthnRequest(<partner="">, <partnerProfile="">, <partnerType="">,
<isPassive="false">, <forceAuthn="false">, <displayOnly="false">,
<delete="false">)
```

Argument	Definition
partner	Indicates the IdP partner to be configured. partner and partnerProfile are exclusive, with one of the two required.
partnerProfile	Indicates the IdP partner profile to be configured. partner and partnerProfile are exclusive, with one of the two required.
partnerType	The type of partner (sp or idp).
isPassive	Indicates if the IdP partner or IdP partner profile should be configured, so that the Authn Request message sent to the IdP will indicate that the IdP should not interact with the user during Federation SSO. True indicates that the IdP should not interact with the user. Optional.
forceAuthn	Indicates if the IdP partner or IdP partner profile should be configured, so that the Authn Request message sent to the IdP will indicate that the IdP should challenge the user even if a valid session exists. True indicates that the user will be challenged. Optional.
displayOnly	Indicates whether or not this command should display the Is Passive and Force Authn settings. Default is false. Optional.
delete	Indicates whether or not this command should delete the Is Passive and Force Authn settings from the specified partner or partner profile. Default is false. Optional.

Example

```
configureIdPAuthnRequest(partner="acme", isPassive="false", forceAuthn="true")
```

configureFedSSOAuthz

A boolean indicating whether or not Authorization for Federation SSO should be enabled.

Description

Enables or disables Authorization for Federation SSO. By default, the authorization feature for Federation SSO will be turned off.

Syntax

```
configureFedSSOAuthz(enabled)
```

Argument	Definition
enabled	Takes as a value true or false.

Example

```
configureFedSSOAuthz("true")
```

configureFedDigitalSignature

Configure the Hashing algorithm used in digital signatures.

Description

If the displayOnly and delete parameters are false, this command will set the algorithm.

Syntax

```
configureFedDigitalSignature(<partner="">,
 <partnerProfile="">, <partnerType="">, <default="false">,
 <algorithm="SHA-256">, <displayOnly="false">, <delete="false">)
```

Argument	Definition
partner	The ID of the SP partner profile
partnerProfile	The Federation Authentication Method to which the Access Manager Authentication Scheme will be mapped
partnerType	The Access Manager Authentication Scheme to which the Federated Authentication Method will be mapped
default	Optional. Boolean indicating whether or not the specified Access Manager Authentication Scheme should be used to challenge the user when the SP requests the Federated Authentication Method
algorithm	Optional. Indicates the authentication level to be used in the mapping in cases when the session authentication level is different from the authentication scheme level
displayOnly	Optional. The application domain in which the underlying policy components will be created
delete	Optional. The HostID used when creating the underlying resource policy object

Example

```
configureFedDigitalSignature(default="true",
 algorithm="SHA-256")
```

configureFedSignEncKey

Configure the signing and/or encryption key alias to be used for digital signature and encryption operations.

Description

Configure the signing and/or encryption key alias to be used for digital signature and encryption operations.

Syntax

```
configureFedSignEncKey(<partner="">, <partnerProfile="">, <partnerType="">,
<default="false">, <signAlias="">, <encAlias="">, <displayOnly="false">,
<delete="false">
```

Argument	Definition
partner	Indicates the partner for which the signing and/or encryption key alias is to be configured. partner, partnerProfile and default parameters are exclusive, with one of the three required
partnerProfile	Indicates the partner profile for which the signing and/or encryption key alias is configured for. partner, partnerProfile and default parameters are exclusive, with one of the three required.
partnerType	Indicates the partner type for which the signing and/or encryption key alias is to be configured. Required when specifying partner or partnerProfile. Valid values are sp or idp.
default	Indicates the global default signing and/or encryption key alias to be configured. partner, partnerProfile and default parameters are exclusive, with one of the three required.
signAlias	The signing key alias. Required when setting the value.
encAlias	The encryption key alias. Required when setting the value.
displayOnly	Indicates whether or not this command should display the signing and encryption key aliases. Default is false. Optional.
delete	Indicates whether or not this command should delete the signing and/or encryption key alias from the specified partner or partner profile. Default is false. Optional.

Example

```
configureFedSignEncKey(default="true", signAlias="osts_signing")
```

configureAttributeSharingSPPartnerNameIDMapping

Configures the NameID to user store attribute mapping to be used during Attribute Sharing.

Description

If displayOnly is true the command displays the NameID to userstore attribute mapping. Else if delete is true the command deletes the specified mapping. Else it sets the enabled flag to the given value and the sets a nameid to userstore attribute mapping.

Syntax

```
configureAttributeSharingSPPartnerNameIDMapping(<partner="">,
 <partnerProfile="">, <enabled="true">, <nameidformat="">,
 <userStoreAttribute="">, <displayOnly="false">, <delete="false">)
```

Argument	Definition
partner	ID of the partner being configured. Optional. partner and partnerProfile parameters are exclusive, with one of the two required.
partnerProfile	Indicates the partner profile for which the mapping is being configured. Optional. partner and partnerProfile parameters are exclusive, with one of the two required
enabled	Boolean indicating if the nameID to userstore attribute mapping is enabled/disabled. Optional. Default value is true.
nameidformat	The NameID format that is mapped to a userStoreAttribute. Optional. Needs to be specified for delete and create/update operations. If not specified for a display operation all the mappings for the specified partner or partnerprofile are displayed. Allowed NameID formats are: ■ orafed-emailaddress for urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress ■ orafed-x509 for urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName ■ orafed-windowsnamequalifier for urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName ■ orafed-kerberos for urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos ■ orafed-transient for urn:oasis:names:tc:SAML:2.0:nameid-format:transient ■ orafed-persistent for urn:oasis:names:tc:SAML:2.0:nameid-format:persistent ■ orafed-unspecified for urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified ■ <customnameidformaturi> for a custom nameid format If the format is set to any other value, the Assertion will be populated with that value.
userStoreAttribute	The userstore attribute to which the specified NameID Format is mapped. Optional. Needs to be specified only for a create or update operation.

Argument	Definition
displayOnly	Indicates whether or not this command should display the NameID to userstore attribute mapping. Default is false. Optional. If set to true the mapping is displayed. If no NameID parameter is specified all the mappings are displayed.
delete	Indicates whether or not this command should delete NameID to userstore attribute mapping. Default is false. Optional.

Examples

```
configureAttributeSharingSPPartnerNameIDMapping(partner="acme",
nameidformat="orafed-emailaddress", userStoreAttribute="mail")

configureAttributeSharingSPPartnerNameIDMapping(partnerProfile="saml20-idp-partner
-profile", nameidformat="orafed-emailaddress", userStoreAttribute="mail")

configureAttributeSharingSPPartnerNameIDMapping(partner="acme")

configureAttributeSharingSPPartnerNameIDMapping(partner="acme", enabled="false")

configureAttributeSharingSPPartnerNameIDMapping(partner="acme",
 displayOnly="true")

configureAttributeSharingSPPartnerNameIDMapping(partner="acme",
 nameidformat="orafed-emailaddress", delete="true")

configureAttributeSharingSPPartnerNameIDMapping(partner="acme",
 nameidformat="orafed-emailaddress", displayOnly="true")
```

configureAttributeSharingIdPPartner

Configures the default attribute sharing nameid and nameid format for the IdP Partner.

Description

Configures the default attribute sharing nameid and nameid format for the IdP Partner.

Syntax

```
configureAttributeSharingIdPPartner(<partner="">,
<partnerProfile="">,<nameidformat="">, <nameidattribute="">)
```

Argument	Definition
partner	ID of the partner being configured. Optional. partner and partnerProfile parameters are exclusive, with one of the two required.
partnerProfile	Indicates the partner profile for which the mapping is being configured. Optional. partner and partnerProfile parameters are exclusive, with one of the two required
nameidformat	The NameID format that is mapped to a userStoreAttribute. Optional. Needs to be specified for delete and create/update operations. If not specified for a display operation all the mappings for the specified partner or partnerprofile are displayed. Allowed NameID formats are: ■ orafed-emailaddress for urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress ■ orafed-x509 for urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName ■ orafed-windowsnamequalifier for urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName ■ orafed-kerberos for urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos ■ orafed-transient for urn:oasis:names:tc:SAML:2.0:nameid-format:transient ■ orafed-persistent for urn:oasis:names:tc:SAML:2.0:nameid-format:persistent ■ orafed-unspecified for urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified ■ orafed-custom for a custom nameid
nameIDAttribute	The attribute in the userstore that should be used as the nameid. Optional.
displayOnly	Indicates whether or not this command should display the NameID to userstore attribute mapping. Default is false. Optional. If set to true the mapping is displayed. If no NameID parameter is specified all the mappings are displayed.

Example

```
configureAttributeSharingIdPPartner(partner="acme",
```

```
nameidformat="orafed-emailaddress", nameidattribute="mail")
```

configureAttributeSharingUserDNToIdPPartnerMapping

Configures Attribute Sharing DN to IdP Mappings.

Description

If displayOnly is set to true the configuration is displayed. If delete is set to true the command deletes a specified mapping; otherwise, a mapping is created or updated.

Syntax

```
configureAttributeSharingUserDNToIdPPartnerMapping(<dn="">,
 <idp="">, <displayOnly="false">, <delete="false">)
```

Argument	Definition
dn	The DN string to map to the given IdP. Optional. Needs to be specified to delete a mapping and set a mapping. If specified for a display operation the mapping for this DN only is displayed.
idp	The partner ID of the IdP to use as Attribute Authority for the given DN. Optional. Needs to be specified only when creating or updating a mapping.
displayOnly	Indicates whether or not this command should display the NameID to userstore attribute mapping. Default is false. Optional. If set to true the mapping is displayed. If no NameID parameter is specified all the mappings are displayed.
delete	Indicates whether or not this command should delete NameID to userstore attribute mapping. Default is false. Optional.

Examples

```
configureAttributeSharingUserDNToIdPPartnerMapping
 (dn="dc=us,dc=oracle, dc=com", displayOnly="true")

configureAttributeSharingUserDNToIdPPartnerMapping(displayOnly="true")

configureAttributeSharingUserDNToIdPPartnerMapping(dn="dc=us,dc=oracle,dc=com",
 delete="true")

configureAttributeSharingUserDNToIdPPartnerMapping(dn="dc=us,dc=oracle,dc=com",
 idp="acme")
```

configureAttributeSharing

Configures the Attribute Sharing feature by setting a default attribute authority.

Description

Configures the Attribute Sharing feature by setting a default attribute authority.

Syntax

```
configureAttributeSharing(<defaultAttributeAuthority="">)
```

Argument	Definition
defaultAttributeAuthority	ID of the partner to use as the default Attribute Authority. Only used when this server is functioning in the SP mode.

Example

```
configureAttributeSharing(defaultAttributeAuthority="acme")

configureAttributeSharing("acme")
```

removeAttributeSharingFromAuthnModule

Removes the Attribute Sharing plug-in from the Authentication Module.

Description

Lists the Federated Authentication Method mappings for the specified Partner.

Syntax

```
removeAttributeSharingFromAuthnModule(<authnModule>, <stepName="">)
```

Argument	Definition
authnModule	The name of the authnModule from which to delete Attribute Sharing plugin.
stepName	The stepName of the Attribute Sharing plugin step to remove. Only needed if there is more than one attribute sharing step. Optional.

Example

```
removeAttributeSharingFromAuthnModule(authnModule="LDAPPlugin")

removeAttributeSharingFromAuthnModule(authnModule="LDAPPlugin",
 stepName="FedAttributeSharing")
```

configureAttributeSharingPlugin

Lists the Federated Authentication Method mappings for a specific Partner Profile.

Description

Configures the input parameters of the Attribute Sharing plugin.

Syntax

```
configureAttributeSharingPlugin(<authnModule>, <stepName=None>,
 <nameIDVariable=None>, <idpVariable=None>, <defaultIdP=None>,
 <nameIDFormatVariable=None>, <defaultNameIDFormat=None>,
 <requestedAttributes=None>)
```

Argument	Definition
authnModule	The name of the authnModule from which to delete Attribute Sharing plugin.
stepName	The stepName of the Attribute Sharing plugin step to remove. Only needed if there is more than one attribute sharing step. Optional.
nameIDVariable	The name of the variable in the session or context that contains the nameID of the user.
idpVariable	The name of the variable in the session or context that contains the idp name to which to send the attribute request.
defaultIdP	The name of the default IdP to send the attribute request to if no IdP can be determined from the session or context.
nameIDFormatVariable	The name of the variable in the session or context that contains the nameID format to use in the attribute request.
defaultNameIDFormat	The default NameID format to use if no nameid format could be determined from the session or context. Allowed NameID formats are: ■ orafed-emailaddress for urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress ■ orafed-x509 for urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName ■ orafed-windowsnamequalifier for urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName ■ orafed-kerberos for urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos ■ orafed-transient for urn:oasis:names:tc:SAML:2.0:nameid-format:transient ■ orafed-persistent for urn:oasis:names:tc:SAML:2.0:nameid-format:persistent ■ orafed-unspecified for urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified If the format is set to any other value, the Assertion will be populated with that value.
requestedAttributes	The attributes to request from the IdP. This string is in the URL query string format.

Example

```
configureAttributeSharingPlugin(authnModule="LDAPPlugin",
 nameIDVariable="dn", idpVariable="attr.idpname", defaultIdP="acme",
 nameIDFormatVariable="attr.nameidformat", defaultNameIDFormat="orafed-x509",
 requestedAttributes="mail&accessAllowed=allowed")
```

insertAttributeSharingInToAuthnModule

Inserts the attribute sharing step into the Authentication Module flow.

Description

Can also be used to remove the attribute sharing step from the Authentication Module flow.

Syntax

```
insertAttributeSharingInToAuthnModule(<authnModule>,
 <fromStep=None>, <fromCond=None>, <toStep=None>, <toCond=None>, <stepName=None>)
```

Argument	Definition
authnModule	The name of the authnModule into which the Attribute Sharing plugin is inserted.
fromStep	The name of the step after which the Attribute Sharing Step (or the step of given name) should be inserted.
fromCond	The condition under which the Attribute Sharing (or step of given name) is called after the fromStep. It has to be one of OnSuccess, OnFailure or OnError.
toStep	The name of the step to go to after the attribute sharing step (or step of given name).
toCond	The condition under which the toStep is called after the Attribute Sharing step (or step of given name).
stepName	The name of the step being added to the flow.

Example

```
insertAttributeSharingInToAuthnModule(authnModule="LDAPPlugin",
 fromStep="stepUA", fromCond="OnSuccess")

insertAttributeSharingInToAuthnModule(authnModule="LDAPPlugin", fromStep="stepUA",
 fromCond="OnSuccess", stepName="success")
```

setSPPartnerAlternateScheme

Provides a way to authenticate clients with an alternate Authentication Scheme.

Description

Identity Federation evaluates an HTTP Header to determine if the alternate Authentication Scheme should be used for this Partner.

Syntax

```
setSPPartnerAlternateScheme(<partner>, <enabled="true">, <httpHeaderName="">,
 <httpHeaderExpression="">, <authnScheme="">, <appDomain="IAM Suite">,
 <hostID="IAMSuiteAgent">, <authzPolicy="Protected Resource Policy">,
 <remove="false">)
```

Argument	Definition
partner	The ID of the partner.
enabled	Indicates whether or not Identity Federation should evaluate the HTTP Header sent by the client
httpHeaderName	Required if enabled is true, the HTTP Header to evaluate. IMPORTANT: This is a global setting and will affect all partners.
httpHeaderExpression	Required if enabled is true, this is the regular expression used to evaluate the value of the HTTP Header.
authnScheme	Required if enabled is true, the alternate Authentication Scheme to be used instead of the default.
appDomain	Optional. The application domain in which the underlying policy components will be created
hostID	Optional. The HostID used when creating the underlying resource policy object
authzPolicy	Optional. The Authorization Policy Name that will be used to protect underlying resource policy object being created.
remove	Optional. If set to true, removes the properties for the alternate scheme in the partner configuration.

Note: ince this operation creates policy objects, it is possible to specify the Application Domain (default: "IAM Suite"), the HostID (default "IAMSuiteAgent") and the Authorization Policy (default "Protected Resource Policy") to be used although the default values can be used.

Example

In this example, Identity Federation is configured to enable the alternate Authentication Scheme at a partner level for the SP partner Acme because the user's browser sends the HTTP Header "User-Agent" with the iPhone string in it. The string triggers the BasicScheme for authentication rather than the default Authentication Scheme.

```
setSPPartnerAlternateScheme("acmeSP", "true", httpHeaderName="User-Agent",
```

```
httpHeaderExpression=".*iPhone.*", authnScheme="BasicScheme")
```

setSPPartnerDefaultScheme

Defines the default Authentication Scheme for the SP partner.

Description

Defines the default Authentication Scheme for the SP partner.

Syntax

```
setSPPartnerDefaultScheme(<partner>, <authnScheme="">, <appDomain="IAM Suite">,
 <hostID="IAMSuiteAgent">, <authzPolicy="Protected Resource Policy">)
```

Argument	Definition
partner	The ID of the partner.
authnScheme	The OAM Authentication Scheme to be used.
appDomain	Optional. The application domain in which the underlying policy components will be created
hostID	Optional. The HostID used when creating the underlying resource policy object
authzPolicy	Optional. The Authorization Policy Name that will be used to protect the underlying resource policy object being created.

Example

```
setSPPartnerDefaultScheme(partnerProfile="acmeSP",
 authnScheme="BasicScheme")
```

setSPPartnerProfileAlternateScheme

Provides a way to authenticate clients with an alternate Authentication Scheme.

Description

Identity Federation evaluates an HTTP Header to determine if the alternate Authentication Scheme should be used for partners assigned to this Partner Profile.

Syntax

```
setSPPartnerProfileAlternateScheme(<partnerProfile>,
 <enabled="true">, <httpHeaderName="">, <httpHeaderExpression="">,
 <authnScheme="">, <appDomain="IAM Suite">, <hostID="IAMSuiteAgent">,
 <authzPolicy="Protected Resource Policy">, <remove="false">)
```

Argument	Definition
partnerProfile	The ID of the partner profile.
enabled	Indicates whether or not Identity Federation should evaluate the HTTP Header sent by the client
httpHeaderName	Required if enabled is true, the HTTP Header to evaluate. IMPORTANT: This is a global setting and will affect all partners.
httpHeaderExpression	Required if enabled is true, this is the regular expression used to evaluate the value of the HTTP Header.
authnScheme	Required if enabled is true, the alternate Authentication Scheme to be used instead of the default.
appDomain	Optional. The application domain in which the underlying policy components will be created
hostID	Optional. The HostID used when creating the underlying resource policy object
authzPolicy	Optional. The Authorization Policy Name that will be used to protect the underlying resource policy object being created.

Note: ince this operation creates policy objects, it is possible to specify the Application Domain (default: "IAM Suite"), the HostID (default "IAMSuiteAgent") and the Authorization Policy (default "Protected Resource Policy") to be used although the default values can be used.

Example

```
setSPPartnerProfileAlternateScheme("acmeSP", "true",
 httpHeaderName="User-Agent", httpHeaderExpression=".*iPhone.*",
 authnScheme="BasicScheme")
```

setSPPartnerProfileDefaultScheme

Sets the default OAM Authentication Scheme to be used to challenge a user for a specific SP Partner Profile.

Description

Sets the default OAM Authentication Scheme to be used to challenge a user for a specific SP Partner Profile.

Syntax

```
setSPPartnerProfileDefaultScheme(<partnerProfile>,
 <authnScheme="">, <appDomain="IAM Suite">, <hostID="IAMSuiteAgent">,
 <authzPolicy="Protected Resource Policy">)
```

Argument	Definition
partnerProfile	The ID of the partner profile.
authnScheme	The OAM Authentication Scheme to be used.
appDomain	Optional. The application domain in which the underlying policy components will be created
hostID	Optional. The HostID used when creating the underlying resource policy object
authzPolicy	Optional. The Authorization Policy Name that will be used to protect the underlying resource policy object being created.

Example

```
setSPPartnerProfileDefaultScheme("saml20-sp-partner-profile",
 "LDAPScheme")
```

addSPPartnerAuthnMethod

Defines a mapping between a Federated Authentication Method and an Access Manager Authentication Scheme for a specific SP Partner.

Description

Maps a Federated Authentication Method to an Access Manager Authentication Scheme for an SP Partner.

Syntax

```
addSPPartnerAuthnMethod(partner, authnMethod, authnScheme,
 isDefault="true", authnLevel="-1", appDomain="IAM Suite",
 hostID="IAMSuiteAgent", <authzPolicy="Protected Resource Policy">)
```

Argument	Definition
partner	The ID of the SP partner.
authnMethod	The Federation Authentication Method to which the Access Manager Authentication Scheme will be mapped
authnScheme	The Access Manager Authentication Scheme to which the Federated Authentication Method will be mapped
isDefault	Optional. Boolean indicating whether or not the specified Access Manager Authentication Scheme should be used to challenge the user when the SP requests the Federated Authentication Method
authnLevel	Optional. Indicates the authentication level to be used in the mapping in cases when the session authentication level is different from the authentication scheme level
appDomain	Optional. The application domain in which the underlying policy components will be created
hostID	Optional. The HostID used when creating the underlying resource policy object
authzPolicy	Optional. The Authorization Policy Name that will be used to protect underlying resource policy object being created.

Example

```
addSPPartnerAuthnMethod("acmeSP",
 "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport",
 "LDAPScheme")
```

addSPPartnerProfileAuthnMethod

Defines a mapping between a Federated Authentication Method to an Access Manager Authentication Scheme for a specific SP Partner Profile.

Description

Maps a Federated Authentication Method to an Access Manager Authentication Scheme for an SP Partner Profile.

Syntax

```
addSPPartnerProfileAuthnMethod(partnerProfile, authnMethod,
 authnScheme, isDefault="true", authnLevel="-1", appDomain="IAM Suite",
 hostID="IAMSuiteAgent", <authzPolicy="Protected Resource Policy">)
```

Argument	Definition
partnerProfile	The ID of the SP partner profile
authnMethod	The Federation Authentication Method to which the Access Manager Authentication Scheme will be mapped
authnScheme	The Access Manager Authentication Scheme to which the Federated Authentication Method will be mapped
isDefault	Optional. Boolean indicating whether or not the specified Access Manager Authentication Scheme should be used to challenge the user when the SP requests the Federated Authentication Method
authnLevel	Optional. Indicates the authentication level to be used in the mapping in cases when the session authentication level is different from the authentication scheme level
appDomain	Optional. The application domain in which the underlying policy components will be created
hostID	Optional. The HostID used when creating the underlying resource policy object
authzPolicy	Optional. The Authorization Policy Name that will be used to protect underlying resource policy object being created.

Example

```
addSPPartnerProfileAuthnMethod("saml20-sp-partner-profile",
  "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport",
  "LDAPScheme")
```

addIdPPartnerAuthnMethod

Sets the Authentication Level to use when creating a session for a Federated Authentication Method for a specific IdP Partner.

Description

Defines the level to which to which users from this IdP partner are authenticated.

Syntax

```
addIdPPartnerAuthnMethod(partner, authnMethod, authnLevel)
```

Argument	Definition
partner	The ID of the SP partner profile
authnMethod	The Federated Authentication Method
authnLevel	The level to use to create the Access Manager user session during a Federation SSO flow for the specified Federated Authentication Method

Example

```
addIdPPartnerAuthnMethod("acmeIdP",
  "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport", "1")
```

addIdPPartnerProfileAuthnMethod

Sets the Authentication Level to use when creating a session for a Federated Authentication Method for a specific IdP Partner Profile.

Description

Defines the level to which to which users from this IdP partner profile are authenticated.

Syntax

```
addIdPPartnerProfileAuthnMethod(partnerProfile, authnMethod,
 authnLevel)
```

Argument	Definition
partnerProfile	The ID of the SP partner profile
authnMethod	The Federated Authentication Method
authnLevel	The level to use to create the Access Manager user session during a Federation SSO flow for the specified Federated Authentication Method

Example

```
addIdPPartnerProfileAuthnMethod("saml20-idp-partner-profile",
  "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport", "1")
```

listPartnerAuthnMethods

Lists the Federated Authentication Method mappings for a specific Partner.

Description

Lists the Federated Authentication Method mappings for the specified Partner.

Syntax

```
listPartnerAuthnMethods(partner, partnerType)
```

Argument	Definition
partner	The ID of the partner
partnerType	The type of the partner (SP or IdP)

Example

```
listPartnerAuthnMethods("acmeSP", "SP")
```

listPartnerProfileAuthnMethods

Lists the Federated Authentication Method mappings for a specific Partner Profile.

Description

Lists the Federated Authentication Method mappings for the specified Partner Profile.

Syntax

```
listPartnerProfileAuthnMethods(partnerProfile, partnerType)
```

Argument	Definition
partnerProfile	The ID of the partner profile
partnerType	The type of the partner (SP or IdP)

Example

```
listPartnerProfileAuthnMethods("saml20-sp-partner-profile", "SP")
```

removePartnerAuthnMethod

Removes the mapping between a Federated Authentication Method and Access Manager Authentication Scheme for a specific Partner.

Description

Removes the mapping between a Federated Authentication Method and Access Manager Authentication Scheme for the specified Partner.

Syntax

```
removePartnerAuthnMethod(<partner>, <partnerType>, <authnMethod>)
```

Argument	Definition
partner	The ID of the partner
partnerType	The type of the partner (SP or IdP)
authnMethod	The Access Manager Authentication Scheme

Example

```
removePartnerAuthnMethod("acmeSP", "SP",
  "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport")
```

removePartnerProfileAuthnMethod

Removes the mapping between a Federated Authentication Method and Access Manager Authentication Scheme for a specific Partner.

Description

Removes the mapping between a Federated Authentication Method and Access Manager Authentication Scheme for the specified Partner.

Syntax

```
removePartnerProfileAuthnMethod(<partnerProfile>,
 <partnerType>, <authnMethod>)
```

Argument	Definition
partnerProfile	The ID of the partner profile
partnerType	The type of the partner (SP or IdP)
authnMethod	The Federated Authentication Method

Example

```
removePartnerProfileAuthnMethod("saml20-sp-partner-profile",
"SP", "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport")
```

setIdPPartnerRequestAuthnMethod

Sets the Federated Authentication Method that will be requested during Federation SSO for a specific IdP Partner.

Description

Sets the Federated Authentication Method that will be requested during Federation SSO for the specified IdP Partner.

Syntax

```
setIdPPartnerRequestAuthnMethod(<partner>, <authnMethod>)
```

Argument	Definition
partner	The ID of the IdP partner
authnMethod	The Federated Authentication Method

Example

```
setIdPPartnerRequestAuthnMethod("acmeIdP",
  "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport")
```

setIdPPartnerProfileRequestAuthnMethod

Sets the Federated Authentication Method that will be requested during Federation SSO for a specific IdP Partner Profile.

Description

Sets the Federated Authentication Method that will be requested during Federation SSO for the specified IdP Partner Profile.

Syntax

```
setIdPPartnerProfileRequestAuthnMethod(<partnerProfile>,
 <authnMethod>)
```

Argument	Definition
partnerProfile	The ID of the IdP partner profile
authnMethod	The Federated Authentication Method

Example

```
setIdPPartnerProfileRequestAuthnMethod("saml20-idp-partner-profile",
  "urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport")
```

useProxiedFedAuthnMethod

Configure the Identity Provider to use the proxied Federation Authentication Method when performing Federation SSO.

Description

If the server acts as an SP with a remote IdP to authenticate the user, when acting as an Identity Provider in a different Federation SSO operation, the server can use the Federation Authentication Method sent by the remote Identity Provider. The server will send the proxied Federation Authentication Method for the list of specified Federation Authentication Schemes. The server will only send the proxied Federation Authentication Method if the Federation protocol used between the server and the Service Provider is the same Federation protocol as the one used between the server and the Identity Provider.

Syntax

```
useProxiedFedAuthnMethod(<enabled="false">,
 <displayOnly="false">, <authnSchemeToAdd="">, <authnSchemeToRemove="">,
 <appDomain="IAM Suite">, <hostID="IAMSuiteAgent">,
 <authzPolicy="Protected Resource Policy">)
```

Argument	Definition
enabled	Indicates whether or not the proxied Federation Authentication Method should be used. Default is to disable the feature. Optional.
displayOnly	Indicates whether or not this command should display the list of Federation Schemes for which the server should send the proxied Federation Authentication Method. Default is false. Optional.
authnSchemeToAdd	The OAM Federation Authentication Scheme to be added to the list of schemes for which the server should send the proxied Federation Authentication Method. authnSchemeToAdd and authnSchemeToRemove parameters are exclusive.
authnSchemeToRemove	The OAM Federation Authentication Scheme to be removed from the list of schemes for which the server should send the proxied Federation Authentication Method. authnSchemeToAdd and authnSchemeToRemove parameters are exclusive.
appDomain	The application domain in which the underlying policy components will be created. Optional.
hostID	The HostID that will be used when creating the underlying resource policy object. Optional.
authzPolicy	Optional. The Authorization Policy Name that will be used to protect the underlying resource policy object being created.

Example

```
useProxiedFedAuthnMethod(enabled="true",
 authnSchemeToAdd="FederationScheme")
```

createFedPartnerProfileFrom

Creates a Federation Partner Profile based on the specified existing one.

Description

Creates a new partner profile based on the specified existing partner profile.

Syntax

```
createFedPartnerProfileFrom(<newPartnerProfile>,
  <existingPartnerProfile>)
```

Argument	Definition
newPartnerProfile	The ID of the new partner profile.
existingPartnerProfile	The ID of the existing partner profile

Example

```
createFedPartnerProfileFrom("newAcmeSPProfile", "acmeSPProfile")
```

deleteFedPartnerProfile

Deletes the specified Federation Partner Profile.

Description

Removes the specified partner profile.

Syntax

```
deleteFedPartnerProfile(<PartnerProfile>)
```

Argument	Definition
PartnerProfile	The ID of the partner profile being deleted.

Example

```
deleteFedPartnerProfile("acmeSPProfile")
```

displayFedPartnerProfile

Displays the properties defined in the specified Federation Partner Profile.

Description

Displays the properties in the specified Federation Partner Profile.

Syntax

```
displayFedPartnerProfile(<PartnerProfile>)
```

Argument	Definition
PartnerProfile	The ID of the partner profile.

Example

```
displayFedPartnerProfile("saml20-idp-partner-profile")
```

listFedPartnerProfiles

Lists all of the existing Federation Partner Profiles.

Description

Lists the existing Federation Partner Profiles.

Syntax

```
listFedPartnerProfiles()
```

This command has no arguments.

Example

```
listFedPartnerProfiles()
```

listFedPartnersForProfile

Lists the partners bound to the specified Federation Partner Profile.

Description

lLists all the partners bound to the specified Federation Partner Profile.

Syntax

```
listFedPartnersForProfile(<PartnerProfile>)
```

Argument	Definition
PartnerProfile	The ID of the partner profile.

Example

```
listFedPartnersForProfile("acmeSPProfile")
```

getFedPartnerProfile

Gets the ID of the Partner Profile bound to the specified partner.

Description

Retrieves the ID of the Partner Profile bound to the specified partner.

Syntax

```
getFedPartnerProfile(<partner>, <partnerType>)
```

Argument	Definition
partner	The ID of the partner.
partnerType	The type of the partner (sp or idp).

Example

```
getFedPartnerProfile("acmeIDP", "idp")
```

setFedPartnerProfile

Sets the Federation Partner Profile ID for the specified partner.

Description

Sets the partner profile for the specified partner profile based on the specified partner profile ID.

Syntax

```
setFedPartnerProfile(<partner>, <partnerType>, <partnerProfile>)
```

Argument	Definition
partner	The ID of the partner.
partnerType	The type of the partner (sp or idp).
partnerProfile	The ID of the partner profile.

Example

```
setFedPartnerProfile("acmeIDP", "idp",
   "saml20-idp-partner-profile")
```

idpinitiatedssoprovideridparam

The value held by `idpinitiatedssoprovideridparam` is used by the peer provider to identify the provider ID of the SP.

Description

Sets the value used to identify the provider ID for the SP.

Syntax

```
updatePartnerProperty(partnerName, partnerType,
   "idpinitiatedssoprovideridparam","providerid", "string")
```

Argument	Definition
partnerName	The ID of the partner
partnerType	Takes as a value either idp or sp
propName	Name of the property being configured or modified
propValue	The value of the property being configured. For an OIF peer IDP, the parameter name must be "providerid". Changing this property will change the parameter name used in the above URL.
type	The data type of the property value. Valid values are string, long, or boolean.

Example

```
updatePartnerProperty(partnerName, "idp",
   "idpinitiatedssoprovideridparam","providerid", "string")
```

idpinitiatedssotargetparam

Sets the target URL for the specified SP partner.

Description

Identifies the target resource. The value held by `idpinitiatedssotargetparam` is used by the peer provider to identify the desired resource; TARGET in the case of Oracle Identity Federation.

Syntax

```
updatePartnerProperty(partnerName, partnerType,
   "idpinitiatedssotargetparam", "TARGET", "string")
```

Argument	Definition
partnerName	The ID of the partner
partnerType	Takes as a value either idp or sp
propName	Name of the property being configured or modified
propValue	The location of the resource. The default value is `TARGET`.
type	The data type of the property value. Valid values are string, long, or boolean.

Example

```
updatePartnerProperty(partnerName, "idp",
   "idpinitiatedssotargetparam", "TARGET", "string")
```

Note: A certificate can be included in a SAML 1.1 signature. By replacing the value of <partnerName> with the partner ID and including the `includecertinsignature` parameter, the certificate will be included with the signature. For example:

```
updatePartnerProperty("<partnerName>", "sp",
 "includecertinsignature", "true", "boolean")

getPartnerProperty("<partnerName>", "sp", "includecertinsignature")

deletePartnerProperty("<partnerName>", "sp",
 "includecertinsignature")
```

6

Mobile and Social WLST Commands

This chapter provides descriptions of custom WebLogic Scripting Tool (WLST) commands for Oracle Access Management Mobile and Social, including command syntax, arguments and examples.

The following section lists the Mobile and Social WLST commands and contains links to the command reference details.

- Mobile and Social Commands

6.1 Mobile and Social Commands

Use the WLST commands listed in Table 6–1 to manage Oracle Access Management Mobile and Social (Mobile and Social) configuration objects.

Table 6–1 WLST Mobile and Social Commands for Mobile Services and Social Identity

Use this command...	To...	Use with WLST...
System Configuration Commands		
getRPSystemConfig	Retrieve system configuration data.	Online
replaceRPSystemConfig	Update system configuration data.	Online
RPApplication Commands		
getRPApplications	Retrieves the RPApplication objects.	Online
removeRPApplication	Deletes the specified RPApplication object.	Online
displayRPApplication	Displays the specified RPApplication object.	Online
createRPApplication	Creates a new RPApplication object.	Online
updateRPApplication	Updates values for a defined RPApplication object.	Online
ServiceProviderInterface Commands		
getServiceProviderInterfaces	Retrieves the RPApplication objects.	Online
removeServiceProviderInterface	Deletes the specified RPApplication object.	Online
displayServiceProviderInterface	Displays the specified RPApplication object.	Online
createServiceProviderInterface	Creates a new RPApplication object.	Online
updateServiceProviderInterface	Updates values for a defined RPApplication object.	Online

Table 6–1 (Cont.) WLST Mobile and Social Commands for Mobile Services and Social

Use this command...	To...	Use with WLST...
Social Identity Provider Commands		
getInternetIdentityProviders	Retrieves the Social Identity Provider objects.	Online
removeInternetIdentityProvider	Deletes the specified Social Identity Provider object.	Online
displayInternetIdentityProvider	Displays the specified Social Identity Provider object.	Online
createInternetIdentityProvider	Creates a new Social Identity Provider object.	Online
updateInternetIdentityProvider	Updates values for a defined Social Identity Provider object.	Online
User Attribute Mapping Commands		
getUserAttributeMappings	Retrieves the User Attribute Mapping objects.	Online
removeUserAttributeMapping	Deletes the specified User Attribute Mapping object.	Online
displayUserAttributeMapping	Displays the specified User Attribute Mapping object.	Online
updateUserAttributeMapping	Updates values for a defined User Attribute Mapping object.	Online
ServiceProvider Commands		
createServiceProvider	Create a ServiceProvider.	Online
updateServiceProvider	Update a ServiceProvider	Online
addRelationshipToServiceProvider	Add a Relationship To a Service Provider.	Online
removeRelationshipFromServiceProvider	Remove a Relationship from a Service Provider.	Online
getServiceProviders	Get a ServiceProvider.	Online
removeServiceProvider	Remove a ServiceProvider object.	Online
displayServiceProvider	Display a ServiceProvider object.	Online
ServiceProfile Commands		
createServiceProfile	Create a service object.	Online
updateServiceProfile	Update a service object.	Online
removeServiceProfile	Remove a service object.	Online
displayServiceProfile	Display a service object.	Online
getServiceProfiles	Retrieve all the service objects.	Online
ApplicationProfile Commands		
getApplicationProfiles	List all ApplicationProfile objects.	Online
createApplicationProfile	Create an ApplicationProfile.	Online
updateApplicationProfile	Update an ApplicationProfile.	Online
removeApplicationProfile	Remove an ApplicationProfile.	Online
displayApplicationProfile	Display an ApplicationProfile.	Online

Table 6–1 (Cont.) WLST Mobile and Social Commands for Mobile Services and Social

Use this command...	To...	Use with WLST...
ServiceDomain Commands		
createServiceDomain	Create a ServiceDomain.	Online
updateServiceDomain	Update a ServiceDomain.	Online
getServiceDomains	Retrieve a ServiceDomain.	Online
removeServiceDomain	Remove a ServiceDomain.	Online
displayServiceDomain	Display a ServiceDomain.	Online
SecurityHandler Commands		
createSecurityHandlerPlugin	Create a SecurityHandlerPlugin.	Online
updateSecurityHandlerPlugin	Update a SecurityHandlerPlugin.	Online
getSecurityHandlerPlugins	Retrieve a SecurityHandlerPlugin.	Online
removeSecurityHandlerPlugin	Remove a SecurityHandlerPlugin.	Online
displaySecurityHandlerPlugin	Display a SecurityHandlerPlugin.	Online
JailBreakingDetectionPolicy Commands		
createJailBreakingDetectionPolicy	Create a JailBreakingDetectionPolicy.	Online
updateJailBreakingDetectionPolicy	Update a JailBreakingDetectionPolicy.	Online
getJailBreakingDetectionPolicys	Retrieve a JailBreakingDetectionPolicy.	Online
removeJailBreakingDetectionPolicy	Remove a JailBreakingDetectionPolicy.	Online
displayJailBreakingDetectionPolicy	Display a JailBreakingDetectionPolicy.	Online

getRPSystemConfig

```
getRPSystemConfig
```

Description

Retrieves the system configuration information.

Syntax

```
getRPSystemConfig( )
```

This command has no arguments.

Example

```
getRPSystemConfig( )
```

replaceRPSystemConfig

replaceRPSystemConfig

Description

Replaces the value of a particular system configuration.

Syntax

```
replaceRPSystemConfig(hostURL, proxyProtocol, proxyHost, proxyPort,
proxyUsername, proxyPassword, attributeList)
```

Argument	Definition
hostURL	The URL of the machine hosting the Mobile and Social server.
proxyProtocol	The proxy protocol (HTTP/HTTPS).
proxyHost	The URL of the proxy machine.
proxyPort	The port of the proxy machine.
proxyUsername	Name of the user accessing the proxy.
proxyPassword	Password of the user accessing the proxy.
attributeList	List of attributes in the JSON format. [{idp:[{name:value},{name:value}],idp2:[{name:value},{name:value}]}]

Example

```
replaceRPSystemConfig('http://515.server.com','http','server.com','18001','proxyUser','proxyPass','[{aas.rest.service:http://adc514:18001/aas_rest}]')
```

getRPApplications

```
getRPApplications
```

Description

Retrieves the RPApplication objects.

Syntax

```
getRPApplications( )
```

This command has no arguments.

Example

```
getRPApplications( )
```

removeRPApplication

```
removeRPApplication
```

Description

Removes the specified RPApplication object.

Syntax

```
removeRPApplication(name)
```

where name is the name of the RPApplication object.

Example

```
removeRPApplication('TestApp')
```

displayRPApplication

```
displayRPApplication
```

Description

Displays the specified RPApplication object.

Syntax

```
displayRPApplication(name)
```

where name is the name of the RPApplication object.

Example

```
displayRPApplication('TestApp')
```

createRPApplication

createRPApplication

Description

Creates a new RPApplication object.

Syntax

```
createRPApplication(identityProviderNameList, sharedSecret, returnUrl,
SPIBindingName, applicationAttributesList, userAttributeMappings,
attributeList, mobileApplicationReturnUrl, name, description)
```

Argument	Definition
identityProviderNameList	A list of Identity Providers
sharedSecret	The shared secret.
returnUrl	The return URL.
SPIBindingName	The SPI binding name.
applicationAttributesList	List of RPApplication attributes specified in the JSON format. [{name1:value1},{name2:value2}]
userAttributeMappings	List of User Attribute Mappings specified in the JSON format. [{idp:[{name:value},{name:value}],idp2:[{name:value},{name:value}]}]
attributeList	List of attributes specified in the JSON format. [{name1:value1},{name2:value2}]
mobileApplicationReturnUrl	The return URL of the mobile application.
name	Name of the object to be created.
description	Description of the object to be created.

Example

```
createRPApplication('Yahoo,Facebook','mySecret','http://me.com',
'OAMServiceProviderInterface','[{pratname1:atval1},{pratname2:atval2}]',
'[{Yahoo:[{uid:email},{mail:email},{zip:postalCode},{country:country}]},
{Facebook:[{uid:email},{mail:email},{zip:postalCode},{country:country}]}]',
'[{atname1:atval2},{atname2:atval2}]','/oam/server','myApp','new Application')
```

updateRPApplication

```
updateRPApplication
```

Description

Updates a particular value for an RPApplication object.

Syntax

```
updateRPApplication(identityProviderNameList, sharedSecret, returnUrl,
SPIBindingName, applicationAttributesList, userAttributeMappings,
attributeList, mobileApplicationReturnUrl, name, description)
```

Argument	Definition
identityProviderNameList	A list of Identity Providers
sharedSecret	The shared secret.
returnUrl	The return URL.
SPIBindingName	The SPI binding name.
applicationAttributesList	List of RPApplication attributes specified in the JSON format. `[{name1:value1},{name2:value2}]`
userAttributeMappings	List of User Attribute Mappings specified in the JSON format. `[{idp:[{name:value},{name:value}],idp2:[{name:value},{name:value}]}]`
attributeList	List of attributes specified in the JSON format. `[{name1:value1},{name2:value2}]`
mobileApplicationReturnUrl	The return URL of the mobile application.
name	Name of the object to be created.
description	Description of the object to be created.

Example

```
updateRPApplication('Facebook,Google','mySecret','http://me.com',
'OAMServiceProviderInterface','[{pratname1:atval1},{pratname2:atval2}]',
'userMap1,userMap2','[{atname1:atval2},{atname2:atval2}]','/oam/server','myApp',
'new Application')
```

getServiceProviderInterfaces

```
getServiceProviderInterfaces
```

Description

Retrieves the Service Provider interface objects.

Syntax

```
getServiceProviderInterfaces( )
```

This command has no arguments.

Example

```
getServiceProviderInterfaces( )
```

removeServiceProviderInterface

```
removeServiceProviderInterface
```

Description

Removes the specified Service Provider interface object.

Syntax

```
removeServiceProviderInterface(name)
```

where name is the name of the Service Provider interface object.

Example

```
removeServiceProviderInterface('TestApp')
```

displayServiceProviderInterface

```
displayServiceProviderInterface
```

Description

Displays the specified Service Provider interface object.

Syntax

```
displayServiceProviderInterface(name)
```

where name is the name of the Service Provider interface object.

Example

```
displayServiceProviderInterface('TestApp')
```

createServiceProviderInterface

createServiceProviderInterface

Description

Creates a new Service Provider interface object.

Syntax

```
createServiceProviderInterface(idpSelectorImpl, postIDPSelectorImpl,
idpInteractionProviderImpl, registrationStatusCheckImpl,
registrationTaskFlowProviderImpl, sessionCreationProviderImpl,
attributeList, name, description)
```

Argument	Definition
idpSelectorImpl	
postIDPSelectorImpl	
idpInteractionProviderImpl	
registrationStatusCheckImpl	
registrationTaskFlowProviderImpl	
sessionCreationProviderImpl	
attributeList	List of attributes in JSON format. `[{idp:[{name:value},{name:value}],idp2:[{name:value},{name:value}]}]`
name	Name of the object to be created.
description	Description of the object to be created.

Example

```
createServiceProviderInterface('idp','postIDP','idpInteraction','regStatus',
'regTask','sessionPro','[{pratname1:atval1},{pratname2:atval2}]','mySPIBind',
'new SPI Binding')
```

updateServiceProviderInterface

updateServiceProviderInterface

Description

Updates a particular value for a Service Provider interface object.

Syntax

```
updateServiceProviderInterface(idpSelectorImpl, postIDPSelectorImpl,
idpInteractionProviderImpl, registrationStatusCheckImpl,
registrationTaskFlowProviderImpl, sessionCreationProviderImpl,
attributeList, name, description)
```

Argument	Definition
idpSelectorImpl	
postIDPSelectorImpl	
idpInteractionProviderImpl	
registrationStatusCheckImpl	
registrationTaskFlowProviderImpl	
sessionCreationProviderImpl	
attributeList	List of attributes in JSON format. `[{idp:[{name:value},{name:value}],idp2:[{name:value},{name:value}]}]`
name	Name of the object to be created.
description	Description of the object to be created.

Example

```
updateServiceProviderInterface('idp','postIDP','idpInteraction','regStatus',
'regTask','sessionPro','[{pratname1:atval1},{pratname2:atval2}]','mySPIBind',
'new SPI Binding')
```

getInternetIdentityProviders

```
getInternetIdentityProviders
```

Description

Retrieves the Social Identity Provider objects.

Syntax

```
getInternetIdentityProviders( )
```

This command has no arguments.

Example

```
getInternetIdentityProviders( )
```

removeInternetIdentityProvider

removeInternetIdentityProvider

Description

Removes the specified Social Identity Provider object.

Syntax

```
removeInternetIdentityProvider(name)
```

where name is the name of the Social Identity Provider object.

Example

```
removeInternetIdentityProvider('TestApp')
```

displayInternetIdentityProvider

```
displayInternetIdentityProvider
```

Description

Displays the specified Social Identity Provider object.

Syntax

```
displayInternetIdentityProvider(name)
```

where name is the name of the Social Identity Provider object.

Example

```
displayInternetIdentityProvider('TestApp')
```

createInternetIdentityProvider

createInternetIdentityProvider

Description

Creates a new Social Identity Provider object.

Syntax

```
createInternetIdentityProvider(icon, protocolType, protocolAttributeList,
providerImplClass, attributeList, name, description)
```

Argument	Definition
icon	Name of the icon.
protocolType	The protocol type is either OpenID, OAuth or Custom.
protocolAttributeList	A list of protocol attributes specified in JSON format. [{name1:value1},{name2:value2}]
providerImplClass	Implementation class for the provider.
attributeList	List of attributes specified in JSON format. [{name1:value1},{name2:value2}]
name	Name of the provider to be created.
description	Description of the provider to be created.

Example

```
createInternetIdentityProvider('myIcon','myType','[{pratname1:atval1},
{pratname2:atval2}]','[{atname1:atval1},{atname2:atval2}]','class','myProvider',
'new Identity Provider')
```

Note: `createInternetIdentityProvider` can also be used within a script to create the provider configuration for Foursquare and Windows Live. The following example is a script for Foursquare. Update the username and password used to connect to the WebLogic Server and the consumer's key and secret values (between the quotes) before executing:

```
url = 't3://localhost:7001'
username='xxxxxx'
password='xxxxxx'
connect(username,password,url)
domainRuntime()

print "Foursquare        OAuth"
print "---------------------"
createInternetIdentityProvider(
    'Foursquare.gif',
    'OAuth',
 '[{oauth.authorization.url:
"https://foursquare.com/oauth2/authorize"},
{oauth.accesstoken.url:
"https://foursquare.com/oauth2/access_token"},
{oauth.profile.url: "https://api.foursquare.com/v2/users/self"},
{oauth.consumer.key:""}, {oauth.consumer.secret:""},
{oauth.rpinstance.name:""}, {oauth.rpinstance.url:""}]',
'[{id:id}, {firstname:firstname}, {lastname:lastname},
{contact.email:contact.email}, {homecity:homecity},
{gender:gender}, {photo:photo}]',
'oracle.security.idaas.rp.oauth.provider.FoursquareImpl',
'Foursquare', 'Foursquare OAuth Provider')

disconnect()
exit()
```

updateInternetIdentityProvider

updateInternetIdentityProvider

Description

Updates a particular value for a Social Identity Provider object.

Syntax

```
updateInternetIdentityProvider(icon, protocolType, protocolAttributeList,
attributeList, providerImplClass, name, description)
```

Argument	Definition
icon	Name of the icon.
protocolType	The protocol type is either OpenID, OAuth or Custom.
protocolAttributeList	A list of protocol attributes specified in JSON format. [{name1:value1},{name2:value2}]
providerImplClass	Implementation class for the provider.
attributeList	List of attributes specified in JSON format. [{name1:value1},{name2:value2}]
name	Name of the provider to be updated.
description	Description of the provider to be updated.

Example

```
updateInternetIdentityProvider('myIcon','myType','[{pratname1:atval1},{pratname2:a
tval2}]','[{atname1:atval1},{atname2:atval2}]','class','myProvider','new Identity
Provider')
```

getUserAttributeMappings

```
getUserAttributeMappings
```

Description

Retrieves the User Attribute Mapping objects.

Syntax

```
getUserAttributeMappings( )
```

This command has no arguments.

Example

```
getUserAttributeMappings( )
```

removeUserAttributeMapping

```
removeUserAttributeMapping
```

Description

Removes the specified User Attribute Mapping object.

Syntax

```
removeUserAttributeMapping(name)
```

where name is the name of the User Attribute Mapping object.

Example

```
removeUserAttributeMapping('TestApp')
```

displayUserAttributeMapping

```
displayUserAttributeMapping
```

Description

Displays the specified User Attribute Mapping object.

Syntax

```
displayUserAttributeMapping(name)
```

where name is the name of the User Attribute Mapping object.

Example

```
displayUserAttributeMapping('TestApp')
```

updateUserAttributeMapping

updateUserAttributeMapping

Description

Updates a particular value for a User Attribute Mapping object.

Syntax

```
updateUserAttributeMapping(application, idp, name,
appProtocolAttributeList)
```

Argument	Definition
application	Name of the application.
idp	Name of the identity provider.
name	Name of the object to be created.
appProtocolAttributeList	List of protocol attributes in JSON format. [{idp:[{name:value},{name:value}],idp2:[{name:value},{name:value}]}]

Example

```
updateUserAttributeMapping('myApp','myProvider','myMap','[{pratname1:atval1},{pratname2:atval2}]')
```

createServiceProvider

createServiceProvider

Description

Creates a Service Provider.

Syntax

```
createServiceProvider(serviceProviderImpl, serviceProviderType,
relationshipList, paramList, name, description)
```

Argument	Definition
serviceProviderImpl	The service provider implementation.
serviceProviderType	The type of service provider. Acceptable values include either Authorization, Authentication, or UserProfile.
relationshipList	The relationship for this Service Provider specified in JSON format: [{relationship:relname,description:descrip,directional1:{name:dirname,description:descrip,providerRelation:relname,entityURIAttrName:uri,scopeAllLevelAttrName:toTop},directional2:{name:dirname,description:descrip,providerRelation:relname,entityURIAttrName:uri,scopeAllLevelAttrName:toTop}}]
paramList	The parameters for this Service Provider specified in JSON format: [{name1:value1},{name2:value2}...]
name	Name of the service provider.
description	Description of the service provider.

Example

```
createServiceProvider('oracle.security.idaas.rest.provider.token.MobileOAMTokenSer
viceProvider', 'Authentication', '[]','[{OAM_VERSION:OAM_11G},{WEBGATE_
ID:accessgate-oic},{ENCRYPTED_PASSWORD:"password"},{DEBUG_VALUE:0},{TRANSPORT_
SECURITY:OPEN},{OAM_SERVER_1:"localhost:5575"},{OAM_SERVER_1_MAX_CONN:4},{OAM_
SERVER_2:"oam_server_2:5575"},{OAM_SERVER_2_MAX_CONN:4}]',
'MobileOAMAuthentication', 'Out Of The Box Mobile Oracle Access Manager (OAM)
 Authentication Service Provider')
```

updateServiceProvider

updateServiceProvider

Description

Updates a Service Provider.

Syntax

```
updateServiceProvider(serviceProviderImpl, serviceProviderType,
relationshipList, paramList, name, description)
```

Argument	Definition
serviceProviderImpl	The service provider implementation
serviceProviderType	The type of service provider - either Authorization, Authentication or UserProfile.
relationshipList	The relationship for this service provider specified in JSON format: [{relationship:relname,description:descrip, directional1:{name:dirname,description:descrip,providerRelation:relname,entityURIAttrName:uri,scopeAllLevelAttrName:toTop},directional2:{name:dirname,description:descrip,providerRelation:relname,entityURIAttrName:uri,scopeAllLevelAttrName:toTop}}]
paramList	The parameters for this Service Provider specified in JSON format: [{name1:value1},{name2:value2}...]
name	Name of the service provider.
description	Description of the service provider.

Example

```
updateServiceProvider('oracle.security.idaas.rest.provider.cruds.ids.
IDSCRUDSServiceProvider', 'UserProfile', '[{relationship:people_groups,
directional1:{name:memberOf, providerRelation:user_memberOfGroup,
entityURIAttrName:person-uri}, directional2:{name:members,
providerRelation:groupMember_user,entityURIAttrName:group-uri }},
{relationship:people_manager, directional1:{name:manager,providerRelation:manager,
entityURIAttrName:report-uri,scopeAllLevelAttrName:toTop},
directional2:{name:reports , providerRelation:reportee,
qntityURIAttrName:manager-uri, scopeAllLevelAttrName:all}},
{relationship:groupMemberOf_groupMembers , directional1:{name:groupMemberOf,
providerRelation:group_memberOfGroup,entityURIAttrName:member-uri},
directional2:{name:groupMembers, providerRelation:groupMember
_group,entityURIAttrName:group-uri }},{relationship:personOwner_ownerOf,
directional1:{name:ownerOf, providerRelation:user_
ownerOfGroup,entityURIAttrName:owner-uri},
directional2:{name:personOwner,providerRelation:groupOwner_
user,entityURIAttrName:group-uri}},{relationship:groupOwner_groupOwnerOf,
directional1:{name:groupOwner, providerRelation:group_
ownerOfGroup,entityURIAttrName:group-uri}, directional2:{name:groupOwnerOf,
providerRelation:groupOwner_group,entityURIAttrName:owner-uri
}}]','[{oracle.ids.name:userrole},{accessControl:false}]', 'UserProfile', 'Out Of
```

```
                The Box User Profile Service Provider')
```

addRelationshipToServiceProvider

addRelationshipToServiceProvider

Description

Adds a Relationship to a Service Provider.

Syntax

```
addRelationshipToServiceProvider(name, relationshipList)
```

Argument	Definition
name	Name of the service provider.
relationshipList	The relationship for this Service Provider specified in JSON format: [{relationship:relname,description:descrip,directional1:{name:dirname,description:descrip,providerRelation:relname,entityURIAttrName:uri,scopeAllLevelAttrName:toTop},directional2:{name:dirname,description:descrip,providerRelation:relname,entityURIAttrName:uri,scopeAllLevelAttrName:toTop}}]

Example

```
addRelationshipToServiceProvider('idsprovider1','[{relationship:relname,
description:descrip, directional1:{name:dirname,description:descrip,
providerRelation:relname,entityURIAttrName:uri,scopeAllLevelAttrName:toTop},
directional2:{name:dirname,description:descrip,
providerRelation:relname,entityURIAttrName:uri,scopeAllLevelAttrName:toTop}}]
```

removeRelationshipFromServiceProvider

```
removeRelationshipFromServiceProvider
```

Description

Removes a Relationship from a Service Provider.

Syntax

```
removeRelationshipFromServiceProvider
```

Argument	Definition
`name`	Name of the service domain.
`relationshipList`	The relationship name for this Service Provider.

Example

```
removeRelationshipFromServiceProvider('idsprovider1','relname')
```

getServiceProviders

```
getServiceProviders
```

Description

Get a service provider.

Syntax

```
getServiceProviders()
```

This command has no arguments.

Example

```
getServiceProviders()
```

The following lines show sample output:

```
ServiceProvider: UserProfile1
ServiceProvider: JWTAuthentication
ServiceProvider: UserProfile
ServiceProvider: MobileOAMAuthentication
ServiceProvider: OAMAuthentication
ServiceProvider: MobileJWTAuthentication
ServiceProvider: sampleauthzserviceprovider
ServiceProvider: InternetIdentityAuthentication
ServiceProvider: OAMAuthorization
```

removeServiceProvider

removeServiceProvider

Description

This command will remove a ServiceProvider object.

Syntax

```
removeServiceProvider(name)
```

where `name` is the name of the ServiceProvider object.

Example

```
removeServiceProvider('name')
```

displayServiceProvider

```
displayServiceProvider
```

Description

This command will display a ServiceProvider object.

Syntax

```
displayServiceProvider(name)
```

where name is the name of the ServiceProvider object.

Example

```
displayServiceProvider('OAMAuthentication')
```

The following lines show sample output:

```
Displaying: ServiceProvider : OAMAuthentication
ReadOnly = 0
Description = Out Of The Box Oracle Access Manager (OAM) Authentication Token
Service Provider
Param = ...
eventProvider = 1
objectName =
com.oracle.idaas:name=OAMAuthentication,type=Xml.ServiceProvider,Xml=MobileService
SystemMBean = 0
ServiceProviderType = Authentication
Name = OAMAuthentication
ConfigMBean = 1
ServiceProviderImpl =
oracle.security.idaas.rest.provider.token.OAMSDKTokenServiceProvider
Relationship = array(javax.management.openmbean.CompositeData,[])
eventTypes = array(java.lang.String,['jmx.attribute.change'])
RestartNeeded = 0
```

createServiceProfile

createServiceProfile

Description

Creates a service.

Syntax

```
createServiceProfile(serviceProvider, supportedTokenList, paramList,
endPoint, name, description, enabled)
```

Argument	Definition
serviceProvider	Name of the service provider.
supportedTokenList	A list of supported tokens specified in JSON format: {type,...} where type is defined as CLIENTTOKEN or USERTOKEN or ACCESSTOKEN or CLIENTREGHANDLE.
paramList	A list of parameters for this Service specified in JSON format: [{name1:value1},{name2:value2}...]
endPoint	The service endpoint.
name	Name of the service.
description	Description of the service.
enabled	Indicates if the service should be enabled or disabled. Boolean flag.

Example

```
createServiceProfile('OAMAuthentication','CLIENTTOKEN,ACCESSTOKEN,USERTOKEN','[]',
'/oamauthentication','OAMAuthentication','Out Of The Box Oracle Access Manager
(OAM) Authentication Token Service',true)
```

updateServiceProfile

updateServiceProfile

Description

Updates a service.

Syntax

```
updateServiceProfile(serviceProvider, supportedTokenList, paramList,
endPoint, name, description, enabled)
```

Argument	Definition
serviceProvider	Name of the service provider.
supportedTokenList	A list of supported tokens specified in JSON format: {type,...} where type is defined as CLIENTTOKEN or USERTOKEN or ACCESSTOKEN or CLIENTREGHANDLE.
paramList	A list of parameters for this Service specified in JSON format: [{name1:value1},{name2:value2}...]
endPoint	The service endpoint.
name	Name of the service.
description	Description of the service.
enabled	Indicates if the service should be enabled or disabled. Boolean flag.

Example

```
updateServiceProfile('MobileJWTAuthentication','CLIENTREGHANDLE,
ACCESSTOKEN,USERTOKEN','[]','/mobilejwtauthentication','MobileJWTAuthentication',
'Out Of The Box Mobile Java Web Token (JWT) Authentication Service Provider',true)
```

removeServiceProfile

```
removeServiceProfile
```

Description

This command will remove a service object.

Syntax

```
removeServiceProfile(name)
```

where `name` is the name of the service to be removed.

Example

```
removeServiceProfile('myService')
```

displayServiceProfile

```
displayServiceProfile
```

Description

This command will display a service object.

Syntax

```
displayServiceProfile(name)
```

where `name` is the name of the service profile to be displayed.

Example

```
displayServiceProfile('OAMAuthorization')
```

The following lines show sample output:

```
Displaying: ServiceProfile : OAMAuthorization
ReadOnly = 0
Enabled = 1
Description = Out Of The Box Oracle Access Manager (OAM) Authorization Service
Provider
Param = array(javax.management.openmbean.CompositeData,[])
eventProvider = 1
SystemMBean = 0
objectName =
com.oracle.idaas:name=OAMAuthorization,type=Xml.ServiceProfile,Xml=MobileService
SupportedToken = array(java.lang.String,[])
ServiceProviderType = Authorization
ServiceProviderName = OAMAuthorization
Name = OAMAuthorization
ConfigMBean = 1
ServiceEndPoint = /oamauthorization
eventTypes = array(java.lang.String,['jmx.attribute.change'])
RestartNeeded = 0
```

getServiceProfiles

```
getServiceProfiles
```

Description

Gets all the service objects.

Syntax

```
getServiceProfiles()
```

This command has no arguments.

Example

```
getServiceProfiles()
```

The following lines show sample output:

```
ServiceProfile: UserProfile1
ServiceProfile: OAMAuthenticatio
ServiceProfile: sampleauthzservice
ServiceProfile: JWTAuthentication
ServiceProfile: UserProfile
ServiceProfile: MobileOAMAuthentication
ServiceProfile: OAMAuthentication
ServiceProfile: MobileJWTAuthentication
ServiceProfile: InternetIdentityAuthentication
ServiceProfile: OAMAuthorization
ServiceProfile: JWTAuthentication1
```

getApplicationProfiles

```
getApplicationProfiles
```

Description

List the ApplicationProfile objects.

Syntax

```
getApplicationProfiles()
```

This command has no arguments.

Example

```
getApplicationProfiles()
```

The following lines show sample output:

```
Contract: MobileExpenseReport1
Contract: MobileAgent2
Contract: MobileBusinessTestApp01
Contract: MobileAgent1
Contract: profileid1
Contract: samplemobileapp2
Contract: profileid2
Contract: samplemobileapp1
```

createApplicationProfile

createApplicationProfile

Description

Creates an ApplicationProfile.

Syntax

```
createApplicationProfile(paramList, mobileAppProfileStr, name,
description)
```

Argument	Definition
paramList	A list of parameters for this Service specified in JSON format: `[{name1:value1},{name2:value2}...]`
mobileAppProfileStr	The mobile app profile string specified in JSON format: `[{clientAppConfigParam:[{name:value},{name:value}], jailBreakingDetectionPolicyName:name}]`
name	Name of the IDaaS Client.
description	Description of the IDaaS Client.

Example

```
createApplicationProfile('[{Mobile.clientRegHandle.baseSecret:welcome1},]',
'[{clientAppConfigParam:[{Mobileparam1:Mobileparam1Value},
{IOSURLScheme:"samplemobileapp1://"},
{AndroidPackage:oracle.android.samplemobileapp1},
{AndroidAppSignature:samplemobileapp1signature}],
jailBreakingDetectionPolicyName:defaultJailBreakingDetectionPolicy}]',
'samplemobileapp1','Sample Mobile App 1')

createApplicationProfile('[{userId4BasicAuth:rest_client1},
{sharedSecret4BasicAuth:"9Qo9olLIl5gDwESYR0hOgw=="},
{signatureAlgorithm:SHA-1}]','','profileid1','OIC Application Profile 1')
```

updateApplicationProfile

updateApplicationProfile

Description

Updates an ApplicationProfile.

Syntax

```
updateApplicationProfile(paramList, mobileAppProfileStr, name,
description)
```

Argument	Definition
paramList	A list of parameters for this Service specified in JSON format: [{name1:value1},{name2:value2}...]
mobileAppProfileStr	The mobile app profile string specified in JSON format: [{clientAppConfigParam:[{name:value},{name:value}], jailBreakingDetectionPolicyName:name}] The value of clientAppConfigParam should match what is defined in the Administration Console on the "Application Profile Configuration Page." Items specified under the 'Configuration Settings' heading are set with the WLST 'clientAppConfigParam'.
name	Name of the IDaaS (Identity as a Service) Client.
description	Description of the IDaaS (Identity as a Service) Client.

Example

```
updateApplicationProfile('[{Mobile.clientRegHandle.baseSecret:welcome1}]','
[{clientAppConfigParam:[{ProfileCacheDuration:60},
{AuthenticationRetryCount:3},{AllowOfflineAuthentication:false},
{ClaimAttributes:"oracle:idm:claims:client:geolocation,
oracle:idm:claims:client:imei,oracle:idm:claims:client:jailbroken,
oracle:idm:claims:client:locale,oracle:idm:claims:client:macaddress,
oracle:idm:claims:client:networktype,oracle:idm:claims:client:ostype,
oracle:idm:claims:client:osversion,oracle:idm:claims:client:phonecarriername,
oracle:idm:claims:client:phonenumber,oracle:idm:claims:client:sdkversion,
oracle:idm:claims:client:udid,oracle:idm:claims:client:vpnenabled"},
{RPWebView:Embedded},{URLScheme:"exp://"},
{IOSBundleID:com.oraclecorp.internal.ExpenseReportApp},
{AndroidAppSignature:"xmlns:xsi=\
'http://www.w3.org/2001/XMLSchema-instance\'
xsi:nil=\'true\'"},{AndroidPackage:"xmlns:xsi=\'
http://www.w3.org/2001/XMLSchema-instance\' xsi:nil=\'true\'"}],
jailBreakingDetectionPolicyName:DefaultJailBreakingDetectionPolicy}]',
'ExpenseApp','OIC Test Expense Sample App')
```

removeApplicationProfile

```
removeApplicationProfile
```

Description

This command removes an ApplicationProfile.

Syntax

```
removeApplicationProfile(name)
```

where `name` is the name of the ApplicationProfile to be removed.

Example

```
removeApplicationProfile('name')
```

displayApplicationProfile

displayApplicationProfile

Description

This command displays the specified ApplicationProfile.

Syntax

dislayApplicationProfile(name)

where name is the name of the ApplicationProfile to be removed.

Example

displayApplicationProfile('MobileAgent1')

The following lines show sample output:

```
Displaying: ApplicationProfile : MobileAgent1
ReadOnly = 0
ConfigMBean = 1
Name = MobileAgent1
MobileAppProfile = None
Description = Mobile Agent App 1
Param =
array(javax.management.openmbean.CompositeData,[javax.management.openmbean.Composi
teDataSupport(compositeType=javax.management.openmbean.CompositeType(name=com.orac
le.xmlns.idm.idaas.idaas_config_11_1_2_0_0.Attribute,items=((itemName=name,
itemType=javax.management.openmbean.SimpleType(name=java.lang.String)),
(itemName=secretValue,itemType=javax.management.openmbean.ArrayType(name=[Ljava.
lang.Character;,dimension=1,elementType=javax.management.openmbean.SimpleType(name
=java.lang.Character),primitiveArray=false)),(itemName=value,itemType=javax.manage
ment.openmbean.SimpleType(name=java.lang.String)))),contents={name=Mobile.reauthnF
orRegNewClientApp, secretValue=null, value=true}),
javax.management.openmbean.CompositeDataSupport(compositeType=javax.management.ope
nmbean.CompositeType(name=com.oracle.xmlns.idm.idaas.idaas_config_11_1_2_0_0.
Attribute,items=((itemName=name,itemType=javax.management.openmbean.SimpleType(nam
e=java.lang.String)),(itemName=secretValue,itemType=javax.management.openmbean.Arr
ayType(name=[Ljava.lang.Character;,dimension=1,elementType=javax.management.openmb
ean.SimpleType(name=java.lang.Character),primitiveArray=false)),(itemName=value,it
emType=javax.management.openmbean.SimpleType(name=java.lang.String)))),contents={n
ame=Mobile.clientRegHandle.baseSecret, secretValue=[Ljava.lang.Character;@11910bd,
value=idaas.ApplicationProfile[MobileAgent1].param[Mobile.clientRegHandle.baseSecr
et]})])
eventProvider = 1
SystemMBean = 0
objectName =
com.oracle.idaas:name=MobileAgent1,type=Xml.ApplicationProfile,Xml=MobileService
eventTypes = array(java.lang.String,['jmx.attribute.change'])
RestartNeeded = 0
```

createServiceDomain

createServiceDomain

Description

Creates a ServiceDomain.

Syntax

```
createServiceDomain(securityHandlerPlugin,serviceBindingList,
clientAppBindingList,mobileAuthStyle,serviceDomainType,name,description)
```

Argument	Definition
securityHandlerPlugin	The name of the securityHandlerPlugin.
serviceBindingList	A list of the ServiceBinding objects in the format: [{serviceName:UserProfile,allowRead:true, allowWrite:true},{serviceName:UserProfile1, allowRead:true,allowWrite:true, requiredToken:[{tokenService:JWTAuthentication, tokenType:{ACCESSTOKEN}}]}, {serviceName:usertokenserviceformobile, requiredToken:[{tokenService:mobilesecurityservice1, tokenType:{ACCESSTOKEN,CLIENTTOKEN}}]}, {serviceName:mobilesecurityservice1}, {serviceName:JWTAuthentication1}, {serviceName:OAMAuthorization}]
clientAppBindingList	A list of client applications specified in the format: [{appName:UserProfile,mobileBinding: [{SSOinclusion:true,SSOpriority:4}]
mobileAuthStyle	Mobile Authentication Style.
serviceDomainType	The type of service domain.
name	Name of the ServiceDomain.
description	Description of the ServiceDomain.

Example

```
createServiceDomain('JunitDebugSecurityHandlerPlugin','[{serviceName:UserProfile,
allowRead:true,allowWrite:true},{serviceName:UserProfile1,allowRead:true,
allowWrite:true,requiredToken:[{tokenService:JWTAuthentication1,
tokenType:ACCESSTOKEN}]},{serviceName:JWTAuthentication},
{serviceName:OAMAuthentication},{serviceName:JWTAuthentication1},
{serviceName:OAMAuthorization,
allowRead:true,allowWrite:false,requiredToken:[{tokenService:OAMAuthentication,
tokenType:USERTOKEN}]}]','[{appName:MobileAgent1,mobileBinding:
[{SSOinclusion:true,SSOpriority:1}]},{appName:MobileBusinessTestApp01,
mobileBinding:[{SSOinclusion:true}]},{appName:MobileAgent2,mobileBinding:
[{SSOinclusion:true,SSOpriority:2}]},{appName:MobileExpenseReport1,
mobileBinding:[{SSOinclusion:false}]},{appName:profileid1}]','','DESKTOP',
'Default','DefaultService Domain ServiceBinding without any requiredToken')
```

updateServiceDomain

```
updateServiceDomain
```

Description

Updates a ServiceDomain.

Syntax

```
updateServiceDomain(securityHandlerPlugin, serviceBindingList,
clientAppBindingList, mobileAuthStyle, serviceDomainType, name,
description)
```

Argument	Definition
`securityHandlerPlugin`	The name of the SecurityHandlerPlugin.
`serviceBindingList`	A list of the ServiceBinding objects in the format: `[{serviceName:UserProfile,allowRead:true, allowWrite:true},{serviceName:UserProfile1, allowRead:true,allowWrite:true, requiredToken:[{tokenService:JWTAuthentication, tokenType:{ACCESSTOKEN}}]}, {serviceName:usertokenserviceformobile, requiredToken:[{tokenService:mobilesecurityservice1, tokenType:{ACCESSTOKEN,CLIENTTOKEN}}]}, {serviceName:mobilesecurityservice1}, {serviceName:JWTAuthentication1}, {serviceName:OAMAuthorization}]`
`clientAppBindingList`	A list of client applications specified in the format: `[{appName:UserProfile,mobileBinding: [{SSOinclusion:true,SSOpriority:4}]`
`mobileAuthStyle`	Mobile Authentication Style.
`serviceDomainType`	The type of Service Domain.
`name`	Name of the ServiceDomain.
`description`	Description of the ServiceDomain.

Example

```
updateServiceDomain('JunitDebugSecurityHandlerPlugin','[{serviceName:UserProfile,
allowRead:true,allowWrite:true},{serviceName:UserProfile1,allowRead:true,
allowWrite:true,requiredToken:[{tokenService:JWTAuthentication1,
tokenType:ACCESSTOKEN}]},{serviceName:JWTAuthentication},
{serviceName:OAMAuthentication},{serviceName:JWTAuthentication1},
{serviceName:OAMAuthorization,allowRead:true,allowWrite:false,
requiredToken:[{tokenService:OAMAuthentication,tokenType:USERTOKEN}]}]',
'[{appName:MobileAgent1,mobileBinding:[{SSOinclusion:true,SSOpriority:1}]},
{appName:MobileBusinessTestApp01,mobileBinding:[{SSOinclusion:true}]},
{appName:MobileAgent2,mobileBinding:[{SSOinclusion:true,SSOpriority:2}]},
{appName:MobileExpenseReport1,mobileBinding:[{SSOinclusion:false}]},
{appName:profileid1}]','','DESKTOP','Default',
```

```
'Default Service Domain ServiceBinding without any requiredToken')
```

getServiceDomains

```
getServiceDomains
```

Description

Get a ServiceDomain.

Syntax

```
getServiceDomains()
```

This command has no arguments.

Example

```
getServiceDomain()
```

The following lines show sample output:

```
ServiceDomain: MobileServiceDomainUTReg
ServiceDomain: MobileRPServiceDomain
ServiceDomain: Contract1
ServiceDomain: MobileJWTServiceDomain
ServiceDomain: MobileRPServiceDomainUTReg
ServiceDomain: MobileContract
ServiceDomain: Default
ServiceDomain: MobileServiceDomain
```

removeServiceDomain

removeServiceDomain

Description

Removes a ServiceDomain.

Syntax

```
removeServiceDomain(name)
```

where `name` is the name of the ServiceDomain to be removed.

Example

```
removeServiceDomain('name')
```

displayServiceDomain

displayServiceDomain

Description

Displays a ServiceDomain.

Syntax

```
displayServiceDomain(name)
```

Example

```
displayServiceDomain('name')
```

The following lines show sample output:

```
Displaying: ServiceDomain : Contract1
ReadOnly = 0
Description = Service Domain 1 using HTTPBasic or Token based Client Token
eventProvider = 1
SystemMBean = 0
objectName =
com.oracle.idaas:name=Contract1,type=Xml.ServiceDomain,Xml=MobileService
MobileAuthStyle = None
ServiceBinding =
array(javax.management.openmbean.CompositeData,[javax.management.openmbean.
CompositeDataSupport(compositeType=javax.management.openmbean.CompositeType(name=
com.oracle.xmlns.idm.idaas.idaas_config_11_1_2_0_0.TServiceBinding,
items=((itemName=allowRead,itemType=javax.management.openmbean.SimpleType(name=
java.lang.Boolean)),(itemName=allowWrite,itemType=javax.management.openmbean.
SimpleType(name=java.lang.Boolean)),(itemName=requiredToken,itemType=javax.managem
ent.openmbean.CompositeType(name=com.oracle.xmlns.idm.idaas.idaas_config_11_1_2_0_
0.TRequiredToken,items=((itemName=tokenService,itemType=javax.management.openmbean
.SimpleType(name=java.lang.String)),(itemName=tokenType,itemType=javax.management.
openmbean.ArrayType(name=[Ljava.lang.String;,dimension=1,elementType=javax.managem
ent.openmbean.SimpleType(name=java.lang.String),primitiveArray=false))))),(itemNam
e=serviceName,itemType=javax.management.openmbean.SimpleType(name=java.lang.String
))))),contents={allowRead=true, allowWrite=true,
requiredToken=javax.management.openmbean.CompositeDataSupport(compositeType=javax.
management.openmbean.CompositeType(name=com.oracle.xmlns.idm.idaas.idaas_config_
11_1_2_0_0.TRequiredToken,
items=((itemName=tokenService,itemType=javax.management.openmbean.SimpleType(name=
java.lang.String)),(itemName=tokenType,itemType=javax.management.openmbean.
ArrayType(name=[Ljava.lang.String;,dimension=1,elementType=javax.management.
openmbean.SimpleType(name=java.lang.String),primitiveArray=false)))),
contents={tokenService=JWTAuthentication, tokenType=[Ljava.lang.String;@d0fbf2}),
serviceName=UserProfile}),
javax.management.openmbean.CompositeDataSupport(compositeType=javax.management.
openmbean.CompositeType(name=
com.oracle.xmlns.idm.idaas.idaas_config_11_1_2_0_0.TServiceBinding,
items=((itemName=allowRead,itemType=javax.management.openmbean.SimpleType(name=
java.lang.Boolean)),(itemName=allowWrite,itemType=javax.management.openmbean.
SimpleType(name=java.lang.Boolean)),(itemName=requiredToken,itemType=
javax.management.openmbean.CompositeType(name=com.oracle.xmlns.idm.idaas.idaas_
config_11_1_2_0_0.TRequiredToken,
items=((itemName=tokenService,itemType=javax.management.openmbean.SimpleType(name=
java.lang.String)),(itemName=tokenType,itemType=javax.management.openmbean.
```

```
ArrayType(name=[Ljava.lang.String;,dimension=1,elementType=javax.management.
openmbean.SimpleType(name=java.lang.String),primitiveArray=false))))),
(itemName=serviceName,itemType=javax.management.openmbean.SimpleType(name=
java.lang.String)))),contents={allowRead=null, allowWrite=null,
requiredToken=null, serviceName=JWTAuthentication})])
MobileCredLevelForRegApp = None
ServiceDomainType = DESKTOP
Name = Contract1
ConfigMBean = 1
ClientAppBinding =
array(javax.management.openmbean.CompositeData,
[javax.management.openmbean.CompositeDataSupport(compositeType=javax.management.
openmbean.CompositeType(name=com.oracle.xmlns.idm.idaas.idaas_config_11_1_2_0_0
TApplicationBinding,items=((itemName=appName,itemType=javax.management.openmbean.
SimpleType(name=java.lang.String)),(itemName=mobileBinding,itemType=javax.
management.openmbean.CompositeType(name=com.oracle.xmlns.idm.idaas.
idaas_config_11_1_2_0_0.TMobileBinding,items=((itemName=SSOinclusion,
itemType=javax.management.openmbean.SimpleType(name=java.lang.Boolean)),
(itemName=SSOpriority,itemType=javax.management.openmbean.SimpleType(name=
java.lang.Short)))))),contents={appName=profileid1, mobileBinding=null}),
javax.management.openmbean.CompositeDataSupport(compositeType=javax.management.
openmbean.CompositeType(name=com.oracle.xmlns.idm.idaas.idaas_config_11_1_2_0_0
.TApplicationBinding,items=((itemName=appName,itemType=javax.management.openmbean
.SimpleType(name=java.lang.String)),(itemName=mobileBinding,itemType=javax.manage
ment.openmbean.CompositeType(name=com.oracle.xmlns.idm.idaas
.idaas_config_11_1_2_0_0.TMobileBinding,items=
((itemName=SSOinclusion,itemType=javax.management.openmbean.SimpleType(name=
java.lang.Boolean)),(itemName=SSOpriority,itemType=javax.management.openmbean.
SimpleType(name=java.lang.Short)))))),contents={appName=profileid2,
mobileBinding=null})])SecurityHandlerPluginName = None
eventTypes = array(java.lang.String,['jmx.attribute.change'])
RestartNeeded = 0
```

createSecurityHandlerPlugin

createSecurityHandlerPlugin

Description

Creates a SecurityHandlerPlugin.

Syntax

```
createSecurityHandlerPlugin(securityHandlerClass, paramList, name,
description)
```

Argument	Definition
securityHandlerClass	Name of the security handler class.
paramList	A list of parameters.
name	Name of the SecurityHandlerPlugin.
description	Description of the SecurityHandlerPlugin.

Example

```
createSecurityHandlerPlugin(
'oracle.security.idaas.rest.provider.plugin.impl.
DefaultMobileSecurityHandlerImpl','
[{allowJailBrokenDevices:false},{requiredHardwareIds:MAC_ADDRESS},
{requiredDeviceProfileAttrs:OSType OSVersion isJailBroken clientSDKVersion}]',
'DefaultSecurityHandlerPlugin','')
```

updateSecurityHandlerPlugin

updateSecurityHandlerPlugin

Description

Updates a SecurityHandlerPlugin.

Syntax

```
updateSecurityHandlerPlugin(securityHandlerClass, paramList, name,
description)
```

Argument	Definition
securityHandlerClass	Name of the security handler class.
paramList	A list of parameters.
name	Name of the SecurityHandlerPlugin.
description	Description of the SecurityHandlerPlugin.

Example

```
updateSecurityHandlerPlugin('oracle.security.idaas.rest.provider.plugin.im
pl.DefaultMobileSecurityHandlerImpl','[{allowJailBrokenDevices:false},{req
uiredHardwareIds:MAC_ADDRESS},{requiredDeviceProfileAttrs:OSType OSVersion
isJailBroken clientSDKVersion}]','DefaultSecurityHandlerPlugin','')
```

getSecurityHandlerPlugins

```
getSecurityHandlerPlugins
```

Description

Gets a SecurityHandlerPlugin.

Syntax

```
getSecurityHandlerPlugins()
```

This command has no arguments.

Example

```
getSecurityHandlerPlugins()
```

The following lines show sample output:

```
SecurityHandlerPlugin: JunitDebugSecurityHandlerPlugin
SecurityHandlerPlugin: OaamSecurityHandlerPlugin
SecurityHandlerPlugin: DefaultSecurityHandlerPlugin
```

removeSecurityHandlerPlugin

removeSecurityHandlerPlugin

Description

Removes a SecurityHandlerPlugin.

Syntax

```
removeSecurityHandlerPlugin(name)
```

where `name` is the name of the SecurityHandlerPlugin to be removed.

Example

```
removeSecurityHandlerPlugin('name')
```

displaySecurityHandlerPlugin

```
displaySecurityHandlerPlugin
```

Description

Displays a SecurityHandlerPlugin.

Syntax

```
displaySecrityHandlerPlugin(name)
```

where `name` is the name of the SecurityHandlerPlugin to be displayed.

Example

```
displaySecurityHandlerPlugin('name')
```

createJailBreakingDetectionPolicy

createJailBreakingDetectionPolicy

Description

Creates a JailBreakingDetectionPolicy.

Syntax

```
createJailBreakingDetectionPolicy(enabled, statementList, name)
```

Argument	Definition
enabled	Enabled.
statementList	A list of parameters.
name	Name of the JailBreakingDetectionPolicy.

Example

```
createJailBreakingDetectionPolicy(true,
'[{minOSVersion:3.5,maxOSVersion:5.0,minClientSDKVersion:1.0,
maxClientSDKVersion:1.0,policyExpirationDurationInSec:3600,
autoCheckPeriodInMin:60,
detectionLocation:[{filePath:"/root",success:true,action:exists},
{filePath:"/opt",success:true,action:exists}]}]',
'defaultJailBreakingDetectionPolicy')
```

updateJailBreakingDetectionPolicy

```
updateJailBreakingDetectionPolicy
```

Description

Updates a JailBreakingDetectionPolicy.

Syntax

```
updateJailBreakingDetectionPolicy(enabled, statementList, name)
```

Argument	Definition
`enabled`	Enabled.
`statementList`	A list of parameters.
`name`	Name of the JailBreakingDetectionPolicy.

Example

```
updateJailBreakingDetectionPolicy(true,'[{minOSVersion:3.5,maxOSVersion:5.
0,minClientSDKVersion:1.0,maxClientSDKVersion:1.0,policyExpirationDuration
InSec:3600,autoCheckPeriodInMin:60,detectionLocation:[{filePath:"/root",su
ccess:true,action:exists},{filePath:"/opt",success:true,action:exists}]}]'
,'defaultJailBreakingDetectionPolicy')
```

getJailBreakingDetectionPolicys

```
getJailBreakingDetectionPolicys
```

Description

Gets the JailBreakingDetectionPolicy.

Syntax

```
getJailBreakingDetectionPolicys()
```

This command has no arguments.

Example

```
getJailBreakingDetectionPolicys()
```

The following lines show sample output:

```
JailBreakingDetectionPolicy: DefaultJailBreakingDetectionPolicy
```

removeJailBreakingDetectionPolicy

```
removeJailBreakingDetectionPolicy
```

Description

Removes a JailBreakingDetectionPolicy.

Syntax

```
removeJailBreakingDetectionPolicy(name)
```

where `name` is the name of the JailBreakingDetectionPolicy.

Example

```
removeJailBreakingDetectionPolicy('name')
```

displayJailBreakingDetectionPolicy

displayJailBreakingDetectionPolicy

Description

Displays a JailBreakingDetectionPolicy.

Syntax

displayJailBreakingDetectionPolicy(name)

where name is the name of the JailBreakingDetectionPolicy.

Example

displayJailBreakingDetectionPolicy('DefaultJailBreakingDetectionPolicy')

The following lines show sample output:

```
Displaying: JailBreakingDetectionPolicy : DefaultJailBreakingDetectionPolicy
ReadOnly = 0
ConfigMBean = 1
Name = DefaultJailBreakingDetectionPolicy
eventProvider = 1
SystemMBean = 0
objectName =
com.oracle.idaas:name=DefaultJailBreakingDetectionPolicy,type=Xml.JailBreakingDete
ctionPolicy,Xml=MobileService
Enable = 1
JailBreakingDetectionPolicyStatement =
array(javax.management.openmbean.CompositeData,[javax.management.openmbean.
CompositeDataSupport(compositeType=javax.management.openmbean.CompositeType(name=
com.oracle.xmlns.idm.idaas.idaas_config_11_1_2_0_0.
TJailBreakingDetectionPolicyStatement,items=((itemName=autoCheckPeriodInMin,
itemType=javax.management.openmbean.SimpleType(name=java.lang.Long)),
(itemName=detectionLocation,itemType=javax.management.openmbean.ArrayType(name=
[Ljavax.management.openmbean.CompositeData;,dimension=1,elementType=
javax.management.openmbean.CompositeType(name=com.oracle.xmlns.idm.idaas.
idaas_config_11_1_2_0_0.
TDetectionLocation,items=((itemName=action,itemType=javax.management.openmbean.
SimpleType(name=java.lang.String)),(itemName=filePath,itemType=javax.management.
openmbean.SimpleType(name=java.lang.String)),(itemName=success,itemType=javax.
management.openmbean.SimpleType(name=java.lang.Boolean)))),primitiveArray=false)),
(itemName=enable,itemType=javax.management.openmbean.SimpleType(name=java.lang.
Boolean)),(itemName=maxClientSDKVersion,itemType=javax.management.openmbean.
SimpleType(name=java.lang.String)),(itemName=maxOSVersion,itemType=javax.
management.openmbean.SimpleType(name=java.lang.String)),(itemName=
minClientSDKVersion,itemType=javax.management.openmbean.SimpleType(name=
java.lang.String)),
(itemName=minOSVersion,itemType=javax.management.openmbean.SimpleType(name=
java.lang.String)),(itemName=policyExpirationDurationInSec,itemType=javax.
management.openmbean.SimpleType(name=java.lang.Long)))),contents=
{autoCheckPeriodInMin=60,detectionLocation=[Ljavax.management.openmbean.
CompositeData;@2dc906,enable=true,maxClientSDKVersion=11.1.2.0.0,
maxOSVersion=null, minClientSDKVersion=11.1.2.0.0, minOSVersion=1.0,
policyExpirationDurationInSec=3600})])
eventTypes = array(java.lang.String,['jmx.attribute.change'])
RestartNeeded = 0
```

7

OAuth Services WLST Commands

This chapter provides descriptions of custom WebLogic Scripting Tool (WLST) commands for Oracle Access Management OAuth Services, including command syntax, arguments and examples.

The following section lists the OAuth Services WLST commands and contains links to the command reference details.

- OAuth Services Commands

7.1 OAuth Services Commands

Use the WLST commands listed in Table 7–1 to manage Oracle Access Management OAuth Services configuration objects.

Table 7–1 WLST Mobile and Social Commands for OAuth Services

Use this command...	To...	Use with WLST...
OAuth Identity Domain Commands		
removeOAuthIdentityDomain	Removes the specified OAuth Identity Domain.	Online
createOAuthIdentityDomain	Creates a new OAuth Identity Domain.	Online
updateOAuthIdentityDomain	Updates an OAuth Identity Domain.	Online
updateOAuthIdentityDomainParam	Updates and allows individual attributes to be modified.	Online
OAuth System Configuration Commands		
updateOAuthSystemConfig	Updates the OAuth System Configuration Defaults for the Identity Domain.	Online
OAuth System Component Commands		
removeOAuthSysComponent	Removes the specified OAuth System Component.	Online
createOAuthSysComponent	Creates the specified OAuth System Component.	Online
updateOAuthSysComponent	Updates the specified OAuth System Component.	Online
OAuth Service Provider Commands		

Table 7–1 (Cont.) WLST Mobile and Social Commands for OAuth Services

Use this command...	To...	Use with WLST...
removeOAuthServiceProvider	This command will remove an OAuth Service Provider object.	Online
createOAuthServiceProvider	Creates an OAuth Service Provider.	Online
updateOAuthServiceProvider	Updates an OAuth Service Provider.	Online
updateOAuthServiceProviderParam	Updates an OAuth Service Provider parameter.	Online
OAuth Client Commands		
removeOAuthClient	Removes an OAuth client object.	Online
createOAuthClient	Creates an OAuth client object.	Online
updateOAuthClient	Updates an OAuth client object.	Online
Service Profile Commands		
removeOAuthServiceProfile	Removes a service profile.	Online
createOAuthServiceProfile	Creates a service profile.	Online
updateOAuthServiceProfile	Updates a service profile.	Online
updateOAuthServiceProfileParam	Updates a service profile and allows individual attributes to be modified.	Online
OAuth Adaptive Access Plug-in Commands		
removeOAuthAdaptiveAccessPlugin	Removes the specified OAuth Adaptive Access Plug-in.	Online
createOAuthAdaptiveAccessPlugin	Creates the specified OAuth Adaptive Access Plug-in.	Online
updateOAuthAdaptiveAccessPlugin	Updates the specified OAuth Adaptive Access Plug-in.	Online
OAuth Token Attributes Plug-in Commands		
removeOAuthTokenAttributesPlugin	Removes the specified OAuth Token Attributes Plug-in.	Online
createOAuthTokenAttributesPlugin	Creates the specified OAuth Token Attributes Plug-in.	Online
updateOAuthTokenAttributesPlugin	Updates the specified OAuth Token Attributes Plug-in.	Online
OAuth ResourceServer Interface Commands		
removeOAuthResourceServerInterface	Removes an OAuth Resource Server Interface.	Online
updateOAuthResourceServerInterface	Updates an OAuth Resource Server Interface.	Online
createOAuthResourceServerInterface	Creates an OAuth Resource Server Interface.	Online
OAuth ResourceServer Interface		
removeOAuthUserProfileResourceServer	Removes an OAuth User Profile Resource Server Interface.	Online

Table 7–1 (Cont.) WLST Mobile and Social Commands for OAuth Services

Use this command...	To...	Use with WLST...
updateOAuthUserProfileResourceServer	Updates an OAuth User Profile Resource Server Interface.	Online
updateOAuthResourceServerInterfaceParam	Updates an OAuth Resource Server Interface and allows an individual attribute to be modified.	Online
createOAuthUserProfileResourceServer	Creates an OAuth User Profile Resource Server Interface.	Online
OAuth MSM Plug-in Commands		
removeOAuthMSMPlugin	Removes the specified OAuth MSM Plugin.	Online
createOAuthMSMPlugin	Creates the specified OAuth MSM Plugin.	Online
updateOAuthMSMPlugin	Updates the specified OAuth MSM Plugin.	Online
updateOAuthMSMPluginParam	Updates an OAuth MSM Plugin.	Online
Get / Display Commands		
getOAuthIdentityDomains	Gets all the existing OAuth Identity Domains.	Online
displayOAuthIdentityDomain	Display the specified OAuth Identity Domain.	Online
displayOAuthSystemConfig	Display the specified OAuth system configuration.	Online
getOAuthSysComponents	Gets all the existing OAuth System Components.	Online
displayOAuthSysComponent	Display the specified OAuth System Component.	Online
getOAuthServiceProviders	Gets all the existing OAuth Service Providers.	Online
displayOAuthServiceProvider	Display the specified OAuth Service Provider.	Online
getOAuthClients	Gets all the existing OAuth Clients.	Online
displayOAuthClient	Display the specified OAuth Client.	Online
getOAuthAdaptiveAccessPlugins	Gets all the existing OAuth AdaptiveAccessPlugins.	Online
displayOAuthAdaptiveAccessPlugin	Display the specified OAuth AdaptiveAccessPlugin.	Online
getOAuthAuthzPlugin	Gets all the existing OAuth authorization plug-ins.	Online
displayOAuthAuthzPlugin	Display the specified OAuth authorization plug-ins.	Online
getOAuthTokenAttributesPlugins	Gets all the existing OAuth Token Attributes Plug-ins.	Online

Table 7–1 (Cont.) WLST Mobile and Social Commands for OAuth Services

Use this command...	To...	Use with WLST...
displayOAuthTokenAttributesPlugin	Display the specified OAuth Token Attributes Plug-in.	Online
getOAuthResourceServerInterfaces	Gets all the existing OAuth ResourceServerInterfaces.	Online
displayOAuthResourceServerInterface	Display the specified OAuth ResourceServerInterface.	Online
getOAuthUserProfileResourceServers	Gets all the existing OAuth UserProfile resource server plug-ins.	Online
displayOAuthUserProfileResourceServer	Display the specified OAuth UserProfile resource server plug-in.	Online
getOAuthServiceProfiles	Gets all the existing OAuth Service Profiles.	Online
displayOAuthServiceProfile	Display the specified OAuth Service Profile.	Online

removeOAuthIdentityDomain

`removeOAuthIdentityDomain`

Description

Removes the specified OAuth Identity Domain.

Syntax

```
removeOAuthIdentityDomain(name)
```

where `name` is the name of the OAuth Identity Domain to be removed.

Example

```
removeOAuthIdentityDomain('myDomain')
```

createOAuthIdentityDomain

```
createOAuthIdentityDomain
```

Description

Creates a new OAuth Identity Domain.

Syntax

```
createOAuthIdentityDomain(name, description, allowMultRS, enableMobile,
globalUID )
```

Argument	Definition
`name`	The name of the OAuth Identity Domain.
`description`	A description of the OAuth Identity Domain. [Optional]
`allowMultRS`	Boolean set for allowing multiple resource servers.
`enableMobile`	Boolean set that enables mobile parameters (used by UI console).
`globalUID`	Global unique identifier. [Optional]

Example

```
createOAuthIdentityDomain('myDomain', 'My Default Identity Domain',
'true', 'true', ' ')
```

updateOAuthIdentityDomain

updateOAuthIdentityDomain

Description

Updates an OAuth Identity Domain.

Syntax

```
updateOAuthIdentityDomain(name, newName, description, allowMultRS,
enableMobile)
```

Argument	Definition
name	The name of the OAuth Identity Domain.
newName	The new name of the OAuth Identity Domain.
description	A description of the OAuth Identity Domain. [Optional]
allowMultRS	Boolean set for allowing multiple resource servers.
enableMobile	Boolean set that enables mobile parameters (used by UI console).

Example

```
updateOAuthIdentityDomain('myDomain','newDomain','My Default Identity
Domain','true','true')
```

updateOAuthIdentityDomainParam

updateOAuthIdentityDomainParam

Description

Updates and allows individual attributes to be modified.

Syntax

```
updateOAuthIdentityDomainParam(name, parameter, newvalue)
```

Argument	Definition
name	The name of the OAuth Identity Domain.
parameter	The parameter to update: name | description | allowTokenAttrRetrieval | enableMobile
new value	The new value for the specified parameter.

Example

```
updateOAuthIdentityDomainParam('myDomain','description','My new
Description')
```

updateOAuthSystemConfig

```
updateOAuthSystemConfig
```

Description

Updates the OAuth system configuration defaults for the identity domain.

Syntax

```
updateOAuthSystemConfig(identityDomainName, proxyProtocol, proxyHost,
proxyPort, proxyUser, minPool, maxPool, keepAlive, maxTokenSearchResult,
paramList )
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`proxyProtocol`	The default HTTP protocol. Either **HTTP** or **HTTPS**. [optional]
`proxyHost`	The default HTTP proxy host. [optional]
`proxyPort`	The default HTTP proxy port. [optional]
`proxyUser`	The default HTTP proxy user. [optional]
`minPool`	The default Apple Push Notification minimum connection pool.
`maxPool`	The default Apple Push Notification maximum connection pool.
`keepAlive`	The default Apple Push Notification keepAlive in seconds.
`maxTokenSearchResult`	The maximum token search result in seconds.
`paramList`	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`

Example

```
updateOAuthSystemConfig('myDomain','HTTP','hostname', '4444', 'user', '1',
'3', '300','55','[{param1:val1},{param2:val2}]')
```

removeOAuthSysComponent

```
removeOAuthSysComponent
```

Description

Removes the specified OAuth system component.

Syntax

```
removeOAuthSysComponent(identityDomainName, name )
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth system component.

Example

```
removeOAuthSysComponent('myDomain','myComponent')
```

createOAuthSysComponent

createOAuthSysComponent

Description

Creates the specified OAuth system component.

Syntax

```
createOAuthSysComponent(identityDomainName, name, description, interClass,
implClass, paramList)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth system component.
description	A description of the OAuth system component. [Optional]
interClass	The interface class of the OAuth system component. ■ **Authorization and consent plug-ins** - oracle.security.idaas.oauth.consent.AuthorizationUserConsent ■ **Client plug-ins** - oracle.security.idaas.oauth.client.ClientSecurityManager ■ **Resource Server Plug-ins** - oracle.security.idaas.oauth.resourceserver.ResourceServerProfileService
implClass	The implement class of the OAuth system component.
paramList	A list of parameters specified in JSON format: [{name1:value1},{name2:value2}...]

Example

```
createOAuthSysComponent('myDomain','DefaultUserConsentService','Default
User Consent
Service','oracle.security.idaas.oauth.consent.AuthorizationUserConsent','o
racle.security.idaas.oauth.consent.impl.LDAPAuthorizationUserConsentImpl',
'[{uc.ldap.username.attr:uid},{uc.ldap.consent.attr:postaladdress},{uc.lda
p.userprofile.service:"/UserProfile"}]')
```

updateOAuthSysComponent

updateOAuthSysComponent

Description

Updates the specified OAuth System Component.

Syntax

```
updateOAuthSysComponent(identityDomainName, name, description, interClass,
implClass, paramList)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth system component.
description	A description of the OAuth system component. [Optional]
interClass	The interface class of the OAuth system component. ■ **Authorization and consent plug-ins** - oracle.security.idaas.oauth.consent.AuthorizationUserConsent ■ **Client plug-ins** - oracle.security.idaas.oauth.client.ClientSecurityManager ■ **Resource Server Plug-ins** - oracle.security.idaas.oauth.resourceserver.ResourceServerProfileService
implClass	The implement class of the OAuth system component.
paramList	A list of parameters specified in JSON format: [{name1:value1},{name2:value2}...]

Example

```
updateOAuthSysComponent('myDomain','DefaultUserConsentService','Default
User Consent
Service','oracle.security.idaas.oauth.consent.AuthorizationUserConsent','o
racle.security.idaas.oauth.consent.impl.LDAPAuthorizationUserConsentImpl',
'[{uc.ldap.username.attr:uid},{uc.ldap.consent.attr:postaladdress},{uc.lda
p.userprofile.service:"/UserProfile"}]')
```

removeOAuthServiceProvider

removeOAuthServiceProvider

Description

Removes an OAuth service provider object.

Syntax

```
removeOAuthServiceProvider(identityDomainName, name )
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth service provider.

Example

```
removeOAuthServiceProvider('myDomain','myProvider')
```

createOAuthServiceProvider

createOAuthServiceProvider

Description

Creates an OAuth service provider

Syntax

```
createOAuthServiceProvider(identityDomainName, name, description,
implClass, paramList)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth service provider.
description	A description of the OAuth service provider. [Optional]
implClass	The implement class of the OAuth service provider.
paramList	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`

Example

```
createOAuthServiceProvider('myDomain','OAuthServiceProvider','OAuth
Service
Provider','oracle.security.idaas.oauth.token.jwtimpl.OAuthProvider',
'[{oam.OAM_VERSION_disabled:OAM_11G},{oam.WEBGATE_
ID:accessgate-oic},{oam.ENCRYPTED_PASSWORD:""},{oam.DEBUG_
VALUE:0},{oam.TRANSPORT_SECURITY:OPEN},{oam.OAM_SERVER_
1:"localhost:5575"},{oam.OAM_SERVER_1_MAX_CONN:4},{oam.OAM_SERVER_2:"oam_
server_2:5575"},{oam.OAM_SERVER_2_MAX_CONN:4},{oam.AuthNURLForUID:"wl_
authen://sample_ldap_no_pwd_protected_res"}]')
```

updateOAuthServiceProvider

updateOAuthServiceProvider

Description

Updates an OAuth service provider.

Syntax

```
updateOAuthServiceProvider(identityDomainName, name, description,
implClass, paramList)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth service provider.
description	A description of the OAuth service provider. [Optional]
implClass	The implement class of the OAuth service provider.
paramList	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`

Example

```
updateOAuthServiceProvider('myDomain','OAuthServiceProvider','OAuth
Service
Provider','oracle.security.idaas.oauth.token.jwtimpl.OAuthProvider',
'[{oam.OAM_VERSION_disabled:OAM_11G},{oam.WEBGATE_
ID:accessgate-oic},{oam.ENCRYPTED_PASSWORD:"welcome"},{oam.DEBUG_
VALUE:0},{oam.TRANSPORT_SECURITY:OPEN},{oam.OAM_SERVER_
1:"localhost:5575"},{oam.OAM_SERVER_1_MAX_CONN:4},{oam.OAM_SERVER_2:"oam_
server_2:5575"},{oam.OAM_SERVER_2_MAX_CONN:4},{oam.AuthNURLForUID:"wl_
authen://sample_ldap_no_pwd_protected_res"}]')
```

updateOAuthServiceProviderParam

updateOAuthServiceProviderParam

Description

Updates a specific parameter with the specified new value.

Syntax

```
updateOAuthServiceProviderParam(identityDomainName, name, param, newvalue)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth service provider.
param	The parameter to update: name, description, implClass, paramList, paramListAdd (adds the specified parameter leaving existing parameters in place)
newvalue	New value for the parameter.

removeOAuthClient

`removeOAuthClient`

Description

Removes an `OAuthClient` object.

Syntax

```
removeOAuthClient(identityDomainName, name )
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth client.

Example

```
removeOAuthClient('myDomain','myClient')
```

createOAuthClient

```
createOAuthClient
```

Description

Creates an `OAuthClient` object.

Syntax

```
createOAuthClient(identityDomainName, name, description, globalUID,
secret, allowTokenAttrRetrieval, httpRedirectURIList, paramList,
mobileRedirectURIList, mobileParams, claimList, minPool, maxPool,
keepAlive, production, gcmAppSetting, scopeRequiresUserConsent,
scopeInvokeUserConsent, allowAllScopes, resourceServerScopes, scopes,
grantTypes, clientType)
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth client.
`description`	A description of the OAuth Client.
`globalUID`	Global unique identifier. [Optional]
`secret`	The secret key.
`allowTokenAttrRetrieval`	Boolean to enable/disable token attribute retrieval.
`httpRedirectList`	The list of one or more redirect URIs specified in JSON format: `[{"uri":partial},{"uri2":partial}...]`
`paramList`	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`
`mobileRedirectURIList`	List of one or more mobile redirect URIs. [Optional]
`mobileParams`	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`
`claimList`	A list of claim attributes. [Optional]
`minPool`	The default Apple Push Notification minimum connection pool. [Optional]
`maxPool`	The default Apple Push Notification maximum connection pool. [Optional]
`keepAlive`	The default Apple Push Notification keepAlive in seconds. [Optional]
`production`	A Boolean to set **production** or **development** mode. [Optional]
`gcmAppSetting`	Google Restricted Package name. [Optional]
`scopeRequiresUserConsent`	Boolean
`scopeInvokeUserConsent`	Boolean
`allowAllScopes`	Boolean

Argument	Definition
resourceServerScopes	List of resource server scopes. Use this argument to select the resource server scope name prefix, for example userProfile would allow a client to access all userProfile resource server scopes. [Optional]
scopes	List of scopes. Use this argument to select a specific scope name, for example: userProfile.me.read. [Optional]
grantTypes	[Optional] List of grant types: ■ authorization_code ■ code ■ token ■ password ■ client_credentials ■ refresh_token ■ oracle-idm:/oauth/grant-type/user-id-assertion
clientType	Type of client: Either CONFIDENTIAL_CLIENT or MOBILE_CLIENT

Example

```
createOAuthClient('myDomain','sampleOAuthMobileClient',
'sample client app','1234567890','quiet','true',
'[{"http://localhost:7005:/base_domain/domainRuntime":false}]','[{par1:val1}]',
'','[{mobpar1:mobval1}]',
'oracle:idm:claims:client:geolocation,oracle:idm:claims:client:imei,
oracle:idm:claims:client:jailbroken,oracle:idm:claims:client:locale,
oracle:idm:claims:client:macaddress,oracle:idm:claims:client:networktype,
oracle:idm:claims:client:ostype,oracle:idm:claims:client:osversion,
oracle:idm:claims:client:phonecarriername,oracle:idm:claims:client:phonenumber,
oracle:idm:claims:client:sdkversion,oracle:idm:claims:client:udid,
oracle:idm:claims:client:vpnenabled,oracle:idm:claims:client:fingerprint',
'1','3','300','false','gcm','true','false','true','','',
'authorization_code,client_credentials','MOBILE_CLIENT')
```

updateOAuthClient

```
updateOAuthClient
```

Description

Updates an OAuthClient.

Syntax

```
updateOAuthClient(identityDomainName, name, description, secret,
allowTokenAttrRetrieval, httpRedirectURIList, paramList,
mobileRedirectURIList, mobileParams, claimList, minPool, maxPool,
keepAlive, production, gcmAppSetting, scopeRequiresUserConsent,
scopeInvokeUserConsent, allowAllScopes, resourceServerScopes, scopes,
grantTypes, clientType)
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth client.
`description`	A description of the OAuth Client.
`secret`	The secret key.
`allowTokenAttrRetrieval`	Boolean to enable/disable token attribute retrieval.
`httpRedirectList`	The list of one or more redirect URIs specified in JSON format: `[{"uri":partial},{"uri2":partial}...]`
`paramList`	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`
`mobileRedirectURIList`	List of one or more mobile redirect URIs. [Optional]
`mobileParams`	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`
`claimList`	A list of claim attributes. [Optional]
`minPool`	The default Apple Push Notification minimum connection pool. [Optional]
`maxPool`	The default Apple Push Notification maximum connection pool. [Optional]
`keepAlive`	The default Apple Push Notification keepAlive in seconds. [Optional]
`production`	A Boolean to set **production** or **development** mode. [Optional]
`gcmAppSetting`	Google Restricted Package name. [Optional]
`scopeRequiresUserConsent`	Boolean
`scopeInvokeUserConsent`	Boolean
`allowAllScopes`	Boolean
`resourceServerScopes`	List of resource server scopes. [Optional]
`scopes`	List of scopes. [Optional]

Argument	Definition
grantTypes	[Optional] List of grant types: ■ authorization_code ■ code ■ token ■ password ■ client_credentials ■ refresh_token ■ oracle-idm:/oauth/grant-type/user-id-assertion
clientType	Type of client: Either CONFIDENTIAL_CLIENT or MOBILE_CLIENT ,ALL

Example

```
updateOAuthClient('myDomain','sampleOAuthMobileClient',
'sample client app','quiet',
'[{"http://localhost:7005:/base_domain/domainRuntime":false}]',
'[{par1:val1}]','','[{mobpar1:mobval1}]','oracle:idm:claims:client:geolocation,
oracle:idm:claims:client:imei,oracle:idm:claims:client:jailbroken,
oracle:idm:claims:client:locale,oracle:idm:claims:client:macaddress,
oracle:idm:claims:client:networktype,oracle:idm:claims:client:ostype,
oracle:idm:claims:client:osversion,oracle:idm:claims:client:phonecarriername,
oracle:idm:claims:client:phonenumber,oracle:idm:claims:client:sdkversion,
oracle:idm:claims:client:udid,oracle:idm:claims:client:vpnenabled,
oracle:idm:claims:client:fingerprint','1','3','300','false','gcm','true','false',
'true','','','authorization_code,client_credentials','MOBILE_CLIENT')
```

updateOAuthClientParam

```
updateOAuthClientParam
```

Description

Updates a specific parameter with the specified new value.

Syntax

```
updateOAuthClient(identityDomainName, name, param, newvalue)
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth client.
`param`	The parameter to update: `[name, description, secret, allowTokenAttrRetrieval, httpRedirectURIList, paramList, paramListAdd` (adds the specified parameter leaving existing parameters in place), `mobileRedirectURIList]`
`newvalue`	New value for the parameter.

Example

```
updateOAuthClientParam('myDomain','sampleOAuthMobileClient','secret',
'xpalkdnwe3')
```

removeOAuthServiceProfile

```
removeOAuthServiceProfile
```

Description

Removes a service profile.

Syntax

```
removeOAuthServiceProfile(identityDomainName, name)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth service profile.

Example

```
removeOAuthServiceProfile('myDomain','myServiceProfile')
```

createOAuthServiceProfile

```
createOAuthServiceProfile
```

Description

Creates a service profile.

Syntax

```
createOAuthServiceProfile(identityDomainName, name, description,
adAccessPlugin, tokenAttrPlugin, clientPlugin, pluginMode,
resourceServerProfilePlugin, authzUserConsentPlugin,
allResourceServerInterfaces, resourceServers, allClients,
clientAppBindings, preferredHardwareIdList, androidSender,
androidSecurityLevel, iosSecurityLevel, otherSecurityLevel,
consentServiceProtection, clientRegRequiresUserConsent, serviceProvider,
endpoint, serviceEnable, mobilePreAuthzExpire, mobilePreAuthzEnable,
authzExpire, authzEnable, clientExpire, clientEnable, clientRefreshExpire,
clientRefreshEnable, userExpire, userEnable, userRefreshExpire,
userRefreshEnable, accessExpire, accessEnable, accessRefreshExpire,
accessRefreshEnable, paramList, mobParamList, userAuthenticator,
tokenStatic, tokenDynamic)
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth system component.
`description`	A description of the OAuth Service Profile. [Optional]
`adAccessPlugin`	Adaptive Access Plug-in. [Optional]
`tokenAttrPlugin`	Token Attribute Plugin. [Optional]
`clientPlugin`	The name of the client plug-in.
`pluginMode`	Client plug-in mode. Either **ALL_LOCAL_STORAGE** or **ALL_PLUGIN_DELEGATION**.
`resourceServerProfilePlugin`	Resource server profile plug-in.
`authzUserConsentPlugin`	Authorization user consent plug-in.
`allResourceServerInterfaces`	Boolean that specifies whether the service profile can contain generic (false) interfaces.
`resourceServers`	List of resource servers.
`allClients`	Boolean that specifies is the service profile applies to all clients.
`clientAppBindings`	[Optional] List of client application bindings specified in JSON format: `[{client:client1,role:SSOAgent,priority:45,param:[{param1:value},{param2:value2}]}]`
`preferredHardwareIdList`	List of Hardware IDs separated by commas.
`androidSender`	GCM sender ID. [Optional]
`androidSecurityLevel`	Android security level: **HIGH** or **MEDIUM** or **LOW**.

Argument	Definition
iosSecurityLevel	iOS security level: **HIGH** or **MEDIUM** or **LOW**.
otherSecurityLevel	Other security level: **HIGH** or **MEDIUM** or **LOW**.
consentServiceProtection	Service Protection Mode: **OAM** or **JWT_IDS** or **JWT_OAM**.
clientRegRequiresUserConsent	Boolean that specifies if client registration requires user consent.
serviceProvider	Service provider.
endpoint	Service endpoint.
serviceEnable	Boolean that enables or disables the service profile. Either **true** or **false**.
mobilePreAuthzExpire	Mobile pre-authorization code expiration length (in seconds). [Optional]
mobilePreAuthzEnable	Boolean that enables or disables the mobile pre-authorization code. [Optional] Either **true** or **false**.
authzExpire	Authorization code expiration (in seconds). [Optional]
authzEnable	Boolean that enables or disables the authorization code. [Optional] Either **true** or **false**.
clientExpire	Client token authorization code expiration (in seconds). [Optional]
clientEnable	Boolean that enables or disables the client token. [Optional] Either **true** or **false**.
clientRefreshExpire	Client refresh token expiration (in seconds). [Optional]
clientRefreshEnable	Boolean that enables or disables the client refresh token. [Optional]
userExpire	User token expiration (in seconds). [Optional]
userEnable	Boolean that enables or disables the user token. [Optional]
userRefreshExpire	User refresh token expiration (in seconds). [Optional]
userRefreshEnable	Boolean that enables or disables the user refresh token. [Optional]
accessExpire	Access token expiration (in seconds).
accessEnable	Boolean access token enable.
accessRefreshExpire	Access refresh token expiration (in seconds).
accessRefreshEnable	Boolean access refresh Token enable.
paramList	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`
mobParamList	A list of mobile client parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`
userAuthenticator	User Authenticator. Either **IDS** or **OAM**.
tokenStatic	[Optional] Static token attribute specified in JSON format: `[{name1:value1},{name2:value2}...]`
tokenDynamic	Dynamic token attribute list. [Optional]

Example

```
createOAuthServiceProfile('myDomain', 'OAuthServiceProfile',
'OAuth Service Profile','sampleSecurityPlugin','defaultTokenAttrPlugin',
'DefaultClientSecurityManager','ALL_LOCAL_STORAGE',
'DefaultResourceServerProfilePlugin','AuthzUserConsentPlugin',
'false','sampleResourceServerInterface','false',
'[{client:sampleOAuthClient,role:SSOAgent,priority:45,param:[{param1:val1},
{param2:val2}]},{client:sampleOwsmOAuthClient,role:SSOAgent,priority:45,
param:[{param1:val1},{param2:val2}]}]','','GoogleCloudMessaging','HIGH','MEDIUM',
'LOW','OAM','true','OAuthServiceProvider','/oauthserv','true','150','false',
'900','true','28800','true','604800','true','28800','true','0','false','3600',
'true','28800','true','[{oracle.id.name:userrole},{jwt.CryptoScheme:RS512},
{jwt.issuer:www.oracle.example.com}]','[{mobileParamName:mobileParamValue}]',
'OAM','[{attr1:val1}]','attr1,attr2,attr3')
```

updateOAuthServiceProfile

updateOAuthServiceProfile

Description

Updates a service profile.

Syntax

```
updateOAuthServiceProfile(identityDomainName, name, description,
adAccessPlugin, tokenAttrPlugin, clientPlugin, pluginMode,
resourceServerProfilePlugin, authzUserConsentPlugin,
allResourceServerInterfaces, resourceServers, allClients,
clientAppBindings, preferredHardwareIdList,androidSender,
androidSecurityLevel, iosSecurityLevel, otherSecurityLevel,
consentServiceProtection, clientRegRequiresUserConsent, serviceProvider,
endpoint, serviceEnable, mobilePreAuthzExpire, mobilePreAuthzEnable,
authzExpire, authzEnable, clientExpire, clientEnable, clientRefreshExpire,
clientRefreshEnable, userExpire, userEnable, userRefreshExpire,
userRefreshEnable, accessExpire, accessEnable, accessRefreshExpire,
accessRefreshEnable, paramList, mobParamList, userAuthenticator,
tokenStatic, tokenDynamic)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth service profile.
description	A description of the OAuth service profile. [Optional]
adAccessPlugin	Adaptive Access Plug-in. [Optional]
tokenAttrPlugin	Token Attribute Plugin. [Optional]
clientPlugin	The name of the client plug-in.
pluginMode	Client plug-in mode. Either **ALL_LOCAL_STORAGE** or **ALL_PLUGIN_DELEGATION**.
resourceServerProfilePlugin	Resource server profile plug-in.
authzUserConsentPlugin	Authorization user consent plug-in.
allResourceServerInterfaces	Boolean that specifies whether the service profile can contain generic (false) interfaces.
resourceServers	List of resource servers.
allClients	Boolean that specifies is the service profile applies to all clients.
clientAppBindings	[Optional] List of client application bindings specified in JSON format: `[{client:client1,role:SSOAgent,priority:45,param:[{param1:value},{param2:value2}]}]`
preferredHardwareIdList	List of Hardware IDs separated by commas.
androidSender	GCM sender ID. [Optional]
androidSecurityLevel	Android security level: **HIGH** or **MEDIUM** or **LOW**.

Argument	Definition
iosSecurityLevel	iOS security level: **HIGH** or **MEDIUM** or **LOW**.
otherSecurityLevel	Other security level: **HIGH** or **MEDIUM** or **LOW**.
consentServiceProtection	Service Protection Mode: **OAM** or **JWT_IDS** or **JWT_OAM**.
clientRegRequiresUserConsent	Boolean that specifies if client registration requires user consent.
serviceProvider	Service provider.
endpoint	Service endpoint.
serviceEnable	Boolean that enables or disables the service profile. Either **true** or **false**.
mobilePreAuthzExpire	Mobile pre-authorization code expiration length (in seconds). [Optional]
mobilePreAuthzEnable	Boolean that enables or disables the mobile pre-authorization code. [Optional] Either **true** or **false**.
authzExpire	Authorization code expiration (in seconds). [Optional]
authzEnable	Boolean that enables or disables the authorization code. [Optional] Either **true** or **false**.
clientExpire	Client token authorization code expiration (in seconds). [Optional]
clientEnable	Boolean that enables or disables the client token. [Optional] Either **true** or **false**.
clientRefreshExpire	Client refresh token expiration (in seconds). [Optional]
clientRefreshEnable	Boolean that enables or disables the client refresh token. [Optional]
userExpire	User token expiration (in seconds). [Optional]
userEnable	Boolean that enables or disables the user token. [Optional]
userRefreshExpire	User refresh token expiration (in seconds). [Optional]
userRefreshEnable	Boolean that enables or disables the user refresh token. [Optional]
accessExpire	Access token expiration (in seconds).
accessEnable	Boolean access token enable.
accessRefreshExpire	Access refresh token expiration (in seconds).
accessRefreshEnable	Boolean access refresh Token enable.
paramList	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`
mobParamList	A list of mobile client parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`
userAuthenticator	User Authenticator. Either **IDS** or **OAM**.
tokenStatic	[Optional] Static token attribute specified in JSON format: `[{name1:value1},{name2:value2}...]`
tokenDynamic	Dynamic token attribute list. [Optional]

Example

```
updateOAuthServiceProfile('myDomain', 'OAuthServiceProfile', 'OAuth
Service
Profile','sampleSecurityPlugin','defaultTokenAttrPlugin','DefaultClientSec
urityManager','ALL_LOCAL_
STORAGE','DefaultResourceServerProfilePlugin','AuthzUserConsentPlugin','fa
lse','sampleResourceServerInterface','false','[{client:sampleOAuthClient,r
ole:SSOAgent,priority:45,param:[{param1:val1},{param2:val2}]},{client:samp
leOwsmOAuthClient,role:SSOAgent,priority:45,param:[{param1:val1},{param2:v
al2}]}]','oracle:idm:claims:client:iosidforvendor,oracle:idm:claims:client
:macaddress,oracle:idm:claims:client:imei','GoogleCloudMessaging','HIGH','
MEDIUM','LOW','OAM','true','OAuthServiceProvider','/oauthserv','true','150
','false','900','true','28800','true','604800','true','28800','true','0','
false','3600','true','28800','true','[{oracle.id.name:userrole},{jwt.Crypt
oScheme:RS512},{jwt.issuer:www.oracle.example.com}]','[{mobileParamName:mo
bileParamValue}]','OAM','[{attr1:val1}]','attr1,attr2,attr3')
```

updateOAuthServiceProfileParam

updateOAuthServiceProfileParam

Description

Updates a specific parameter with the specified new value.

Syntax

```
updateOAuthServiceProfileParam(domainName, name, parameter, newvalue)
```

Argument	Definition
domainName	The name of the OAuth identity domain.
name	The name of the OAuth service profile.
parameter	The parameter to update: name | description | adAccessPlugin | msmPlugin | tokenAttrPlugin | clientPlugin | pluginMode | resourceServerProfilePlugin | authzUserConsentPlugin | allResourceServerInterfaces | resourceServers | allClients | clientAppBindings | androidSender | androidSecurityLevel | iosSecurityLevel | otherSecurityLevel | consentServiceProtection | clientRegRequiresUserConsent | serviceProvider | endpoint | serviceEnable | mobilePreAuthzExpire | mobilePreAuthzEnable | authzExpire | authzEnable | clientExpire | clientEnable | clientRefreshExpire | clientRefreshEnable | userExpire | userEnable | userRefreshExpire | userRefreshEnable | accessExpire | accessEnable | accessRefreshExpire | accessRefreshEnable | paramList | paramListAdd | mobParamList | userAuthenticator | tokenStatic | tokenDynamic | preferredHardwareIdList | preferredHardwareIdListAdd
newvalue	New value for the specified parameter.

Example

```
updateOAuthServiceProfileParam('myDomain', 'OAuthServiceProfile',
'description','My new Description')
```

removeOAuthAdaptiveAccessPlugin

```
removeOAuthAdaptiveAccessPlugin
```

Description

Removes the specified OAuth Adaptive Access plug-in.

Syntax

```
removeOAuthAdaptiveAccessPlugin(identityDomainName,name)
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth system component.

Example

```
removeOAuthAdaptiveAccessPlugin('myDomain','myComponent')
```

createOAuthAdaptiveAccessPlugin

```
createOAuthAdaptiveAccessPlugin
```

Description

Creates the specified OAuth Adaptive Access plug-in.

Syntax

```
createOAuthAdaptiveAccessPlugin(identityDomainName, name, description,
implClass, paramList)
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth plug-in.
`description`	A description of the OAuth plug-in. [Optional]
`implClass`	The implement class of the OAuth plug-in.
`paramList`	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`

Example

```
createOAuthAdaptiveAccessPlugin('myDomain','sampleSecurityPlugin','sample
adaptive access plugin',
'oracle.security.idaas.rest.provider.plugin.impl.DebugMobileSecurityHandle
rImpl','[{OAUTH_TEST:true},{EMU_DEVICE_REG:true},{EMU_HANDLE:false}]')
```

updateOAuthAdaptiveAccessPlugin

updateOAuthAdaptiveAccessPlugin

Description

Updates the specified OAuth Adaptive Access plug-in.

Syntax

```
updateOAuthAdaptiveAccessPlugin(identityDomainName, name, description,
implClass, paramList)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth plug-in.
description	A description of the OAuth plug-in. [Optional]
implClass	The implement class of the OAuth plug-in.
paramList	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`

Example

```
updateOAuthAdaptiveAccessPlugin('myDomain','sampleSecurityPlugin','sample
adaptive access
plugin','oracle.security.idaas.rest.provider.plugin.impl.DebugMobileSecuri
tyHandlerImpl','[{OAUTH_TEST:true},{EMU_DEVICE_REG:true},{EMU_
HANDLE:false}]')
```

removeOAuthTokenAttributesPlugin

```
removeOAuthTokenAttributesPlugin
```

Description

Removes the specified OAuth Token Attributes plug-in.

Syntax

```
removeOAuthTokenAttributesPlugin(identityDomainName,name)
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth system component.

Example

```
removeOAuthTokenAttributesPlugin('myDomain','myComponent')
```

createOAuthTokenAttributesPlugin

```
createOAuthTokenAttributesPlugin
```

Description

Creates the specified OAuth Token Attributes plug-in.

Syntax

```
createOAuthTokenAttributesPlugin(identityDomainName, name,
description,implClass, paramList)
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth plug-in.
`description`	A description of the OAuth plug-in. [Optional]
`implClass`	The implement class of the OAuth plug-in.
`paramList`	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`

Example

```
createOAuthTokenAttributesPlugin('myDomain','testTokenAttributesPlugin','t
est token attributes
plugin','oracle.security.idaas.rest.provider.plugin.impl.DebugTokenAttribu
tesHandlerImpl','[{paramName:paramValue}]')
```

updateOAuthTokenAttributesPlugin

```
updateOAuthTokenAttributesPlugin
```

Description

Updates the specified OAuth Token Attributes plug-in.

Syntax

```
updateOAuthTokenAttributesPlugin(identityDomainName, name, description,
implClass, paramList)
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth plug-in.
`description`	A description of the OAuth plug-in. [Optional]
`implClass`	The implement class of the OAuth plug-in.
`paramList`	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`

Example

```
updateOAuthTokenAttributesPlugin('myDomain','testTokenAttributesPlugin','t
est token attributes plugin',
'oracle.security.idaas.rest.provider.plugin.impl.DebugTokenAttributesHandl
erImpl','[{paramName:paramValue}]')
```

removeOAuthResourceServerInterface

removeOAuthResourceServerInterface

Description

Removes an OAuth resource server interface.

Syntax

```
removeOAuthResourceServerInterface(identityDomainName, name )
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth resource server interface.

Example

```
removeOAuthResourceServerInterface('myDomain','myComponent')
```

updateOAuthResourceServerInterface

updateOAuthResourceServerInterface

Description

Updates an OAuth resource server interface.

Syntax

```
updateOAuthResourceServerInterface(identityDomainName, name, description,
secret, allowTokenAttrRetrieval, namespacePrefix, audienceClaim,
scopeList, offlineScope, authzUserConsentPluginRef, overriddenAuthzExpire,
overriddenAuthzEnable, overriddenAccessExpire, overriddenAccessEnable,
overriddenAccessRefreshExpire, overriddenAccessRefreshEnable, tokenStatic,
tokenDynamic)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth resource server interface.
description	A description of the OAuth resource server interface.
secret	The secret key.
allowTokenAttrRetrieval	Boolean that enables/disables token attribute retrieval.
namespacePrefix	A namespace prefix. [Optional]
audienceClaim	Audience claim URI. [Optional]
scopeList	A list of parameters specified in JSON format: `[{scopeName:myName,includedInDefault:true,userOffline:true,requiresConsent:true,scopeDesc:[{en-us:value},{en:value2}]}]`
offlineScope	Offline scope. [Optional]
authzUserConsentPluginRef	Authorization UserConsent plug-in reference.
overriddenAuthzExpire	Overridden authorization code expiration (in seconds).
overriddenAuthzEnable	Boolean that enables/disables the authorization override option.
overriddenAccessExpire	Overridden access token expiration (in seconds).
overriddenAccessEnable	Boolean that enables/disables the access token override option.
overriddenAccessRefreshExpire	Overridden access refresh token expiration (in seconds).
overriddenAccessRefreshEnable	Boolean that enables/disables the access refresh override option.
tokenStatic	A list of static token attributes specified in JSON format: `[{name1:value1},{name2:value2}...]`
tokenDynamic	Dynamic token attribute list.

Example

```
updateOAuthResourceServerInterface('myDomain','sampleResourceServerInterfa
ce','sample portal content resource
server','secret','true','namespaceprefix.','audienceClaim','[{scopeName:sa
mplePortalContentServer.portal.read,includedInDefault:false,userOffline:fa
lse,requiresConsent:true,scopeDesc:[{en-us:read portal
content}]},{scopeName:samplePortalContentServer.portal.write,includedInDef
ault:false,userOffline:false,requiresConsent:true,scopeDesc:[{en-us:write
portal
content}]}]','offlineScope','AuthzUserConsentPlugin','1200','false','7200'
,'false','28801','false','[]','')
```

updateOAuthResourceServerInterfaceParam

updateOAuthResourceServerInterfaceParam

Description

Updates a specific parameter with the specified new value.

Syntax

```
updateOAuthResourceServerInterfaceParam(identityDomainName, name, param,
newvalue)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth resource server interface.
param	The parameter to update: [name, description, secret, allowTokenAttrRetrieval, namespacePrefix, audienceClaim, scopeList, offlineScope, authzUserConsentPluginRef].
newvalue	New value for the parameter.

Example

```
updateOAuthResourceServerInterfaceParam('myDomain',
'sampleResourceServerInterface','namespacePrefix','xyz.')
```

createOAuthResourceServerInterface

createOAuthResourceServerInterface

Description

Creates an OAuth resource server interface.

Syntax

```
createOAuthResourceServerInterface(identityDomainName, name, description,
globalUID, secret, allowTokenAttrRetrieval, namespacePrefix,
audienceClaim, scopeList, offlineScope, authzUserConsentPluginRef,
overriddenAuthzExpire, overriddenAuthzEnable, overriddenAccessExpire,
overriddenAccessEnable, overriddenAccessRefreshExpire,
overriddenAccessRefreshEnable, tokenStatic, tokenDynamic)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth resource server interface.
description	A description of the OAuth resource server interface.
globalUID	Global unique identifier. [Optional]
secret	The secret key.
allowTokenAttrRetrieval	Boolean that enables/disables token attribute retrieval.
namespacePrefix	A namespace prefix. [Optional]
audienceClaim	Audience claim URI. [Optional]
scopeList	A list of parameters specified in JSON format: `[{scopeName:myName,includedInDefault:true,userOffline:true,requiresConsent:true,scopeDesc:[{en-us:value},{en:value2}]}]`
offlineScope	Offline scope. [Optional]
authzUserConsentPluginRef	Authorization UserConsent plug-in reference.
overriddenAuthzExpire	Overridden authorization code expiration (in seconds).
overriddenAuthzEnable	Boolean that enables/disables the authorization override option.
overriddenAccessExpire	Overridden access token expiration (in seconds).
overriddenAccessEnable	Boolean that enables/disables the access token override option.
overriddenAccessRefreshExpire	Overridden access refresh token expiration (in seconds).
overriddenAccessRefreshEnable	Boolean that enables/disables the access refresh override option.
tokenStatic	A list of static token attributes specified in JSON format: `[{name1:value1},{name2:value2}...]`
tokenDynamic	Dynamic token attribute list.

Example

```
createOAuthResourceServerInterface('myDomain','sampleResourceServerInterface',
'sample portal content resource server','','secret','true','namespaceprefix.',
'audienceClaim','[{scopeName:samplePortalContentServer.portal.read,
includedInDefault:false,userOffline:false,requiresConsent:true,
scopeDesc:[{en-us:read portal content}]},
{scopeName:samplePortalContentServer.portal.write,
includedInDefault:false,userOffline:false,requiresConsent:true,
scopeDesc:[{en-us:write portal content}]}]',
'offlineScope','AuthzUserConsentPlugin','1200','false','7200','false','28801',
'false','[]','')
```

removeOAuthUserProfileResourceServer

```
removeOAuthUserProfileResourceServer
```

Description

Removes an OAuth User Profile resource server interface.

Syntax

```
removeOAuthUserProfileResourceServer(identityDomainName, name)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth user profile resource server interface.

Example

```
removeOAuthUserProfileResourceServer('myDomain','myComponent')
```

updateOAuthUserProfileResourceServer

```
updateOAuthUserProfileResourceServer
```

Description

Updates an OAuth User Profile resource server interface.

Syntax

```
updateOAuthUserProfileResourceServer(identityDomainName, resName, resDesc,
secret, namespacePrefix, authzPluginRef, scopeList, offlineScope,
authzExpire, authzEnable, accessExpire, accessEnable, accessRefreshExpire,
accessRefreshEnable, tokenStatic, tokenDynamic, endpoint, enabled,
subResource, paramList)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
resName	The name of the OAuth resource server interface.
resDesc	A description of the OAuth resource server interface.
secret	The secret key.
namespacePrefix	A namespace prefix.
authzPluginRef	Authorization plug-in reference.
scopeList	A list of parameters specified in JSON format: `[{scopeName:myName,includedInDefault:true,userOffline:true,requiresConsent:true,scopeDesc:[{en-us:value},{en:value2}]}]`
offlineScope	Offline scope. [Optional]
authzExpire	Authorization code expiration (in seconds)
authzEnable	Boolean that enables/disables the authorization code option.
accessExpire	Access token expiration (in seconds).
accessEnable	Boolean that enables/disables the access token option.
accessRefreshExpire	Access refresh token expiration (in seconds).
accessRefreshEnable	Boolean that enables/disables the access refresh option.
tokenStatic	A list of static token attributes specified in JSON format: `[{name1:value1},{name2:value2}...]`
tokenDynamic	Dynamic token attribute list.
endpoint	Service endpoint.
enabled	Boolean to enable/disable.

Argument	Definition
subResource	Specified in JSON format: [{endpoint:"/sub",enabled:true,implClass:com.oracle.impl, entities:[{attributes:"at1,at2", relationship:[{ name:people_groups,endpoint:memberOf, srcEntity:person-uri,destEntity:group-uri}]}], binding:[{method:GET,allow:true,scope:myscope,addScope:[{name:newscope,attr:"var1,var2"}], param:[{param1:val1}]}]}]
paramList	A list of parameters specified in JSON format: [{name1:value1},{name2:value2}...]

Example

```
 updateOAuthUserProfileResourceServer('myDomain','userProfile',
'Out Of The Box User Profile Resource Server','welcome1',
'[{scopeName:userProfile.users.read,includedInDefault:false,userOffline:false,
requiresConsent:false,scopeDesc:[{en-us:read any user default profile}]},
{scopeName:userProfile.users.write,includedInDefault:false,userOffline:false,
requiresConsent:false,scopeDesc:[{en-us:write any user default profile}]},
{scopeName:userProfile.group.read,includedInDefault:false,userOffline:false,
requiresConsent:false,scopeDesc:[{en-us:read any group default profile}]},
{scopeName:userProfile.group.write,includedInDefault:false,userOffline:false,
requiresConsent:false,scopeDesc:[{en-us:write any group default profile}]},
{scopeName:userProfile.me.read,includedInDefault:false,userOffline:false,
requiresConsent:true,scopeDesc:[{en-us:read my default profile}]},
{scopeName:userProfile.me.write,includedInDefault:false,userOffline:false,
requiresConsent:true,scopeDesc:[{en-us:write my default profile}]},
{scopeName:userProfile.me.password,includedInDefault:false,userOffline:false,
requiresConsent:true,scopeDesc:[{en-us:write my default password}]}]',
'namespace','userrole','defaultPlugin','900','true','604800','true','28800',
'true','/myuserprofile','false','[{accessControl:false},
{adminGroup:"cn=Administrators,ou=groups,ou=myrealm,dc=base_domain"},
{selfEdit:true}]','[{endpoint:"/me",
enabled:true,implClass:oracle.security.idaas.oauth.jaxrs.Me,
entities:[{attributes:"",relationship:[{name:people_groups,
endpoint:memberOf,srcEntity:person-uri,destEntity:group-uri,scopeNames:""},
{name:people_manager,endpoint:manager,srcEntity:report-uri,
destEntity:manager-uri,scopeNames:""}]}],binding:[{method:"GET",
allow:true,scope:myscope,addScope:[{name:userProfile.me.read,attr:"uid,mail,
description,commonname,firstname,lastname"},{name:userProfile.me.password,
attr:password}]},{method:"POST,PUT,DELETE",allow:true,scope:myscope,
addScope:[{name:userProfile.me.write,attr:"uid,mail,description,commonname,
firstname,lastname"},{name:userProfile.me.password,attr:password}]}],param:[]},
{endpoint:"/users",enabled:true,implClass:oracle.
security.idaas.oauth.jaxrs.Users,entities:[{attributes:"",
relationship:[{name:people_groups,endpoint:memberOf,srcEntity:person-uri,
destEntity:group-uri,scopeNames:""},{name:people_manager,endpoint:manager,
srcEntity:report-uri,destEntity:manager-uri,scopeNames:""}]}],
binding:[{method:"GET",allow:true,scope:myscope,
addScope:[{name:userProfile.users.read,attr:"uid,mail,description,commonname,
firstname,lastname"}]},{method:"POST,PUT,DELETE",allow:true,scope:myscope,
addScope:[{name:userProfile.users.write,attr:"uid,mail,description,commonname,
firstname,lastname"}]}],param:[]},{endpoint:"/groups",enabled:true,
```

```
implClass:oracle.security.idaas.oauth.jaxrs.Groups,entities:[{attributes:"",
relationship:[{name:groups_people,endpoint:memberOf,srcEntity:group-uri,
destEntity:person-uri,scopeNames:""}]}],binding:[{method:"GET",allow:true,
scope:myscope,addScope:[{name:userProfile.group.read,attr:"name,description"}]},
{method:"POST,PUT,DELETE",allow:true,scope:myscope,
addScope:[{name:userProfile.group . write,attr:"name,description"}]}],
param:[]}]','[{param1:val1},{param2:val2}]','attr1,attr2')
```

updateOAuthResourceServerInterfaceParam

updateOAuthResourceServerInterfaceParam

Description

Updates an OAuth resource server interface and allows individual attributes to be modified.

Syntax

```
updateOAuthResourceServerInterfaceParam(domainName, name, parameter,
newvalue)
```

Argument	Definition
domainName	The name of the OAuth identity domain.
name	The name of the OAuth resource server interface.
parameter	The parameter to update: name | description | allowTokenAttrRetrieval | secret | namespacePrefix
newvalue	The new value for the specified parameter.

Example

```
updateOAuthResourceServerInterfaceParam('myDomain','Resource','description','My
new Description')
```

createOAuthUserProfileResourceServer

```
createOAuthUserProfileResourceServer
```

Description

Creates an OAuth User Profile resource server interface.

Syntax

```
createOAuthUserProfileResourceServer(identityDomainName, resName, resDesc,
globalUID, secret, scopeList, namespacePrefix, idsName, authzPluginRef,
authzExpire, authzEnable, accessExpire, accessEnable, accessRefreshExpire,
accessRefreshEnable, endpoint, enabled, paramList, subResourceList,
tokenStatic, tokenDynamic)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
resName	The name of the OAuth resource server interface.
resDesc	A description of the OAuth resource server interface.
globalUID	Global unique identifier. (Optional)
secret	The secret key.
scopeList	A list of parameters specified in JSON format: `[{scopeName:myName,includedInDefault:true,userOffline:true,requiresConsent:true,scopeDesc:[{en-us:value},{en:value2}]}]`
namespacePrefix	A namespace prefix.
idsName	The identity directory service name.
authzPluginRef	Authorization plug-in reference.
authzExpire	Authorization code expiration (in seconds)
authzEnable	Boolean that enables/disables the authorization code option.
accessExpire	Access token expiration (in seconds).
accessEnable	Boolean that enables/disables the access token option.
accessRefreshExpire	Access refresh token expiration (in seconds).
accessRefreshEnable	Boolean that enables/disables the access refresh option.
endpoint	Service endpoint.
enabled	Boolean to enable/disable.
paramList	A list of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`

Argument	Definition
subResource	Specified in JSON format: [{endpoint:"/sub",enabled:true,implClass:com.oracle.impl, entities:[{attributes:"at1,at2", relationship:[{ name:people_groups,endpoint:memberOf, srcEntity:person-uri,destEntity:group-uri}]}], binding:[{method:GET,allow:true,scope:myscope,addScope:[{name:newscope,attr:"var1,var2"}], param:[{param1:val1}]}]}]
tokenStatic	A list of static token attributes specified in JSON format: [{name1:value1},{name2:value2}...]
tokenDynamic	Dynamic token attribute list.

Example

```
createOAuthUserProfileResourceServer('myDomain','userProfile',
'Out Of The Box User Profile Resource Server','555888','welcome1',
'[{scopeName:userProfile.users.read,includedInDefault:false,userOffline:false,
requiresConsent:false,scopeDesc:[{en-us:read any user default profile}]},
{scopeName:userProfile.users.write,includedInDefault:false,userOffline:false,
requiresConsent:false,scopeDesc:[{en-us:write any user default profile}]},
{scopeName:userProfile.group.read,includedInDefault:false,userOffline:false,
requiresConsent:false,scopeDesc:[{en-us:read any group default profile}]},
{scopeName:userProfile.group.write,includedInDefault:false,userOffline:false,
requiresConsent:false,scopeDesc:[{en-us:write any group default profile}]},
{scopeName:userProfile.me.read,includedInDefault:false,userOffline:false,
requiresConsent:true,scopeDesc:[{en-us:read my default profile}]},
{scopeName:userProfile.me.write,includedInDefault:false,userOffline:false,
requiresConsent:true,scopeDesc:[{en-us:write my default profile}]},
{scopeName:userProfile.me.password,includedInDefault:false,userOffline:false,
requiresConsent:true,scopeDesc:[{en-us:write my default password}]}]',
'namespace','userrole','defaultPlugin','900','true','604800','true','28800',
'true','/myuserprofile','false','[{accessControl:false},
{adminGroup:"cn=Administrators,ou=groups,ou=myrealm,dc=base_domain"},
{selfEdit:true}]','[{endpoint:"/me",enabled:true,
implClass:oracle.security.idaas.oauth.jaxrs.Me,entities:[{attributes:"",
relationship:[{name:people_groups,endpoint:memberOf,srcEntity:person-uri,
destEntity:group-uri,scopeNames:""},{name:people_manager,endpoint:manager,
srcEntity:report-uri,destEntity:manager-uri,scopeNames:""}]}],
binding:[{method:"GET",allow:true,scope:myscope,
addScope:[{name:userProfile.me.read,attr:"uid,mail,description,commonname,
firstname,lastname"},{name:userProfile.me.password,attr:password}]},
{method:"POST,PUT,DELETE",allow:true,scope:myscope,
addScope:[{name:userProfile.me.write,attr:"uid,mail,description,commonname,
firstname,lastname"},{name:userProfile.me.password,attr:password}]}],param:[]},
{endpoint:"/users",enabled:true,
implClass:oracle.security.idaas.oauth.jaxrs.Users,entities:[{attributes:"",
relationship:[{name:people_groups,endpoint:memberOf,srcEntity:person-uri,
destEntity:group-uri,scopeNames:""},{name:people_manager,endpoint:manager,
srcEntity:report-uri,destEntity:manager-uri,scopeNames:""}]}],
binding:[{method:"GET",allow:true,scope:myscope,
addScope:[{name:userProfile.users.read,attr:"uid,mail,description,commonname,
firstname,lastname"}]},{method:"POST,PUT,DELETE",allow:true,scope:myscope,
addScope:[{name:userProfile.users.write,attr:"uid,mail,description,commonname,
```

```
firstname,lastname"}]}],param:[]},{endpoint:"/groups",enabled:true,
implClass:oracle.security.idaas.oauth.jaxrs.Groups,
entities:[{attributes:"",relationship:[{name:groups_people,endpoint:memberOf,
srcEntity:group-uri,destEntity:person-uri,scopeNames:""}]}],
binding:[{method:"GET",allow:true,scope:myscope,
addScope:[{name:userProfile.group.read,attr:"name,description"}]},
{method:"POST,PUT,DELETE",allow:true,scope:myscope,
addScope:[{name:userProfile.group.write,attr: "name,description"}]}],
param:[]}]','[{param1:val1},{param2:val2}]','attr1,attr2')
```

removeOAuthMSMPlugin

`removeOAuthMSMPlugin`

Description

Removes the specified OAuth MSM Plug-in.

Syntax

```
removeOAuthMSMPlugin(identityDomainName, name )
```

Argument	Definition
`identityDomainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth system component.

Example

```
removeOAuthMSMPlugin('myDomain','myComponent')
```

createOAuthMSMPlugin

```
createOAuthMSMPlugin
```

Description

Creates the specified OAuth MSM Plug-in.

Syntax

```
createOAuthMSMPlugin(identityDomainName, name, description, implClass,
paramList)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth plug-in.
description	Description of the OAuth plug-in.
implClass	Implement class of the OAuth Plug-in.
paramList	List of parameters specified in JSON format: `[{name1:value1},{name2:value2}...]`

Example

```
createOAuthMSMPlugin('myDomain','sampleSecurityPlugin','sample msm
plugin','oracle.security.idaas.rest.provider.plugin.impl.DebugMobileSecuri
tyHandlerImpl','[{OAUTH_TEST:true},{EMU_DEVICE_REG:true},{EMU_
HANDLE:false}]')
```

updateOAuthMSMPlugin

updateOAuthMSMPlugin

Description

Updates the specified OAuth MSM Plug-in.

Syntax

```
updateOAuthMSMPlugin(identityDomainName, name, description, implClass,
paramList)
```

Argument	Definition
identityDomainName	The name of the OAuth identity domain.
name	The name of the OAuth plug-in.
description	Description of the OAuth plug-in.
implClass	Implement class of the OAuth Plug-in.
paramList	List of parameters specified in JSON format: [{name1:value1},{name2:value2}...]

Example

```
updateOAuthMSMPlugin('myDomain','sampleSecurityPlugin','sample msm
plugin','oracle.security.idaas.rest.provider.plugin.impl.DebugMobileSecuri
tyHandlerImpl','[{OAUTH_TEST:true},{EMU_DEVICE_REG:true},{EMU_
HANDLE:false}]')
```

updateOAuthMSMPluginParam

```
updateOAuthMSMPluginParam
```

Description

Updates an OAuth MSM Plug-in.

Syntax

```
updateOAuthMSMPluginParam(domainName, name, parameter, newvalue)
```

Argument	Definition
`domainName`	The name of the OAuth identity domain.
`name`	The name of the OAuth plug-in.
`parameter`	Parameter to update: `name` | `description` | `implClass` | `paramList` | `paramListAdd` | `paramListUpdate` | `paramListRemove`
`newvalue`	New value for the specified parameter.

Example

```
updateOAuthMSMPluginParam('myDomain','defaultMSMPlugin','description','My
new Description')
```

getOAuthIdentityDomains

```
getOAuthIdentityDomains
```

Description

Gets all the existing OAuth identity domains.

Syntax

```
getOAuthIdentityDomains()
```

Example

```
getOAuthIdentityDomains()
```

displayOAuthIdentityDomain

displayOAuthIdentityDomain

Description

Display the specified OAuth identity domain.

Syntax

```
displayOAuthIdentityDomain(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
displayOAuthIdentityDomain('12345678-1234-1234-1234-123456789012')
```

displayOAuthSystemConfig

```
displayOAuthSystemConfig
```

Description

Display the specified OAuth system configuration.

Syntax

```
displayOAuthSystemConfig(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
displayOAuthSystemConfig('12345678-1234-1234-1234-123456789012')
```

getOAuthSysComponents

```
getOAuthSysComponents
```

Description

Gets all the existing OAuth system components.

Syntax

```
getOAuthSysComponents(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
getOAuthSysComponents('12345678-1234-1234-1234-123456789012')
```

displayOAuthSysComponent

```
displayOAuthSysComponent
```

Description

Display the specified OAuth system component.

Syntax

```
displayOAuthSysComponent(uuid,name)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.
name	The name of the specific system component.

Example

```
displayOAuthSysComponent('12345678-1234-1234-1234-123456789012','DefaultTo
kenLifeCycleService')
```

getOAuthServiceProviders

```
getOAuthServiceProviders
```

Description

Gets all the existing OAuth service providers.

Syntax

```
getOAuthServiceProviders(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
getOAuthServiceProviders('12345678-1234-1234-1234-123456789012')
```

displayOAuthServiceProvider

```
displayOAuthServiceProvider
```

Description

Display the specified OAuth service provider.

Syntax

```
displayOAuthServiceProvider(uuid,name)
```

Argument	Definition
`uuid`	The universally unique identifier for the identity domain.
`name`	The name of the specific service provider.

Example

```
displayOAuthServiceProvider('12345678-1234-1234-1234-123456789012','OAuthS
erviceProvider')
```

getOAuthClients

```
getOAuthClients
```

Description

Gets all the existing OAuth Clients.

Syntax

```
getOAuthClients(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
getOAuthClients('12345678-1234-1234-1234-123456789012')
```

displayOAuthClient

```
displayOAuthClient
```

Description

Display the specified OAuth client.

Syntax

```
displayOAuthClient(uuid,name)
```

Argument	Definition
`uuid`	The universally unique identifier for the identity domain.
`name`	The name of the specific client.

Example

displayOAuthClient('12345678-1234-1234-1234-123456789012','sampleOAuthClient')

getOAuthAdaptiveAccessPlugins

```
getOAuthAdaptiveAccessPlugins
```

Description

Gets all the existing OAuth Adaptive Access plug-ins.

Syntax

```
getOAuthAdaptiveAccessPlugins(uuid)
```

Argument	Definition
`uuid`	The universally unique identifier for the identity domain.

Example

```
getOAuthAdaptiveAccessPlugins('12345678-1234-1234-1234-123456789012')
```

displayOAuthAdaptiveAccessPlugin

```
displayOAuthAdaptiveAccessPlugin
```

Description

Display the specified OAuth adaptive access plug-in.

Syntax

```
displayOAuthAdaptiveAccessPlugin(uuid,name)
```

Argument	Definition
`uuid`	The universally unique identifier for the identity domain.
`name`	The name of the specific adaptive access plug-in.

Example

```
displayOAuthAdaptiveAccessPlugin('12345678-1234-1234-1234-123456789012','s
ampleOAuthClient')
```

getOAuthAuthzPlugin

```
getOAuthAuthzPlugin
```

Description

Gets all the existing OAuth authorization plug-ins..

Syntax

```
getOAuthAuthzPlugin(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
getOAuthAuthzPlugin('12345678-1234-1234-1234-123456789012')
```

displayOAuthAuthzPlugin

```
displayOAuthAuthzPlugin
```

Description

Display the specified OAuth authorization plug-in.

Syntax

```
displayOAuthAuthzPlugin(uuid,name)
```

Argument	Definition
`uuid`	The universally unique identifier for the identity domain.
`name`	The name of the specific authorization plug-in.

Example

```
displayOAuthAuthzPlugin('12345678-1234-1234-1234-123456789012','sampleOAut
hClient')
```

getOAuthTokenAttributesPlugins

```
getOAuthTokenAttributesPlugins
```

Description

Gets all the existing OAuth token attributes plug-ins.

Syntax

```
getOAuthTokenAttributesPlugins(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
getOAuthTokenAttributesPlugins('12345678-1234-1234-1234-123456789012')
```

displayOAuthTokenAttributesPlugin

```
displayOAuthTokenAttributesPlugin
```

Description

Display the specified OAuth token attributes plug-in.

Syntax

```
displayOAuthTokenAttributesPlugin(uuid,name)
```

Argument	Definition
`uuid`	The universally unique identifier for the identity domain.
`name`	The name of the specific token attributes plug-in.

Example

```
displayOAuthTokenAttributesPlugin('12345678-1234-1234-1234-123456789012','
sampleOAuthClient')
```

getOAuthResourceServerInterfaces

```
getOAuthResourceServerInterfaces
```

Description

Gets all the existing OAuth resource server interfaces.

Syntax

```
getOAuthResourceServerInterfaces(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
getOAuthResourceServerInterfaces('12345678-1234-1234-1234-123456789012')
```

displayOAuthResourceServerInterface

displayOAuthResourceServerInterface

Description

Display the specified OAuth resource server interface.

Syntax

displayOAuthResourceServerInterface(uuid,name)

Argument	Definition
uuid	The universally unique identifier for the identity domain.
name	The name of the specific resource server interface.

Example

```
displayOAuthResourceServerInterface('12345678-1234-1234-1234-123456789012'
,'sampleOAuthClient')
```

getOAuthUserProfileResourceServers

```
getOAuthUserProfileResourceServers
```

Description

Gets all the existing OAuth User Profile resource server plug-ins.

Syntax

```
getOAuthUserProfileResourceServers(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
getOAuthUserProfileResourceServers('12345678-1234-1234-1234-123456789012')
```

displayOAuthUserProfileResourceServer

```
displayOAuthUserProfileResourceServer
```

Description

Display the specified OAuth User Profile resource server plug-in.

Syntax

```
displayOAuthUserProfileResourceServer(uuid,name)
```

Argument	Definition
`uuid`	The universally unique identifier for the identity domain.
`name`	The name of the specific User Profile resource server plug-in.

Example

```
displayOAuthUserProfileResourceServer('12345678-1234-1234-1234-12345678901
2','UserProfile')
```

getOAuthServiceProfiles

```
getOAuthServiceProfiles
```

Description

Gets all the existing OAuth service profiles.

Syntax

```
getOAuthServiceProfiles(uuid)
```

Argument	Definition
uuid	The universally unique identifier for the identity domain.

Example

```
getOAuthServiceProfiles('12345678-1234-1234-1234-123456789012')
```

displayOAuthServiceProfile

```
displayOAuthServiceProfile
```

Description

Display the specified OAuth service profile.

Syntax

```
displayOAuthServiceProfile(uuid,name)
```

Argument	Definition
`uuid`	The universally unique identifier for the identity domain.
`name`	The name of the specific service profile.

Example

```
displayOAuthServiceProfile('12345678-1234-1234-1234-123456789012','OAuthSe
rviceProfile')
```

8

Security Token Service WLST Commands

This chapter provides descriptions of custom WebLogic Scripting Tool (WLST) commands for Oracle Access Management Security Token Service, including command syntax, arguments and examples.

The following section lists the Security Token Service WLST commands and contains links to the command reference details.

- Security Token Service Commands

8.1 Security Token Service Commands

The Oracle Access Management Security Token Service (Security Token Service) WLST commands are divided into the following categories.

- **Partner Commands** are related to tasks involving partners.
- **WS-Prefix to Relying Party Partner Mapping Commands** are used to map a service URL, specified in the **AppliesTo** field of a WS-Trust RST request, to a partner of type Relying Party. The WS prefix string can be an exact service URL, or a URL with a parent path to the service URL. For example, if a mapping is defined to map a WS Prefix (http://test.com/service) to a Relying Party (RelyingPartyPartnerTest), then the following service URLs would be mapped to the Relying Party: http://test.com/service, http://test.com/service/calculatorService, http://test.com/service/shop/cart...
- **Partner Profiles Commands** are related to tasks involving partner profiles.
- **Issuance Templates Commands** are related to tasks involving issuance templates.
- **Validation Templates Commands** are related to tasks involving validation templates.

Table 8–1 is divided into five sections and describes the various WLST commands in each of these categories. Use the WLST commands listed to manage the Security Token Service.

Table 8–1 WLST Commands for Security Token Service

Use this command...	To...	Use with WLST...
Partner Commands		
getPartner	Retrieve a partner and print result.	Online
getAllRequesterPartners	Retrieve the names of Requester partners.	Online

Table 8–1 (Cont.) WLST Commands for Security Token Service

Use this command...	To...	Use with WLST...
getAllRelyingPartyPartners	Retrieve the names of all Relying Party partners.	Online
getAllIssuingAuthorityPartners	Retrieve the names of all Issuing Authority partners.	Online
isPartnerPresent	Query Security Token Service to determine whether or not the partner exists in the Partner store.	Online
createPartner	Create a new Partner entry.	Online
updatePartner	Update an existing Partner entry based on the provided information.	Online
deletePartner	Delete a partner entry.	Online
getPartnerUsernameTokenUsername	Retrieve the partner's username value.	Online
getPartnerUsernameTokenPassword	Retrieve the partner's password value.	Online
setPartnerUsernameTokenCredential	Set the username and password values of a partner entry.	Online
deletePartnerUsernameTokenCredential	Remove the username and password values from a partner entry.	Online
getPartnerSigningCert	Retrieve the Base64 encoded signing certificate for the partner.	Online
getPartnerEncryptionCert	Retrieve the Base64 encoded encryption certificate for the partner.	Online
setPartnerSigningCert	Upload the signing certificate to the partner entry.	Online
setPartnerEncryptionCert	Upload the encryption certificate to the partner entry.	Online
deletePartnerSigningCert	Remove the signing certificate from the partner entry.	Online Offline
deletePartnerEncryptionCert	Remove the encryption certificate from the partner entry.	Online Offline
getPartnerAllIdentityAttributes	Retrieve and display all Identity mapping attributes used to map a token to a requester partner.	Online Offline
getPartnerIdentityAttribute	Retrieve and display the identity mapping attribute.	Online Offline
setPartnerIdentityAttribute	Set the identity mapping attribute for a requester partner.	Online Offline
deletePartnerIdentityAttribute	Delete the identity mapping attribute for a requester partner.	Online Offline
Relying Party Partner Mapping Commands		
getAllWSPrefixAndPartnerMappings	Retrieve and display all WS Prefixes.	Online Offline

Table 8–1 (Cont.) WLST Commands for Security Token Service

Use this command...	To...	Use with WLST...
getWSPrefixAndPartnerMapping	Retrieve and display the Relying Party Partner mapped to the specified wsprefix parameter.	Online Offline
createWSPrefixAndPartnerMapping	Create a new WS Prefix mapping to a Relying Partner.	Online Offline
deleteWSPrefixAndPartnerMapping	Delete an existing WS Prefix mapping to a Relying Partner.	Online Offline
Partner Profiles Commands		
getAllPartnerProfiles	Retrieve the names of all the existing partner profiles.	Online
getPartnerProfile	Retrieve partner profile configuration data.	Online
createRequesterPartnerProfile	Create a new Requester Partner profile with default configuration data.	Online
createRelyingPartyPartnerProfile	Create a new Relying Party Partner profile with default configuration data.	Online
createIssuingAuthorityPartnerProfile	Create a new Issuing Authority Partner profile with default configuration data.	Online
deletePartnerProfile	Delete an existing partner profile.	Online
Issuance Template Commands		
getAllIssuanceTemplates	Retrieve the names of all the existing Issuance Templates.	Online Offline
getIssuanceTemplate	Retrieve configuration data of a specific Issuance Template.	Online
createIssuanceTemplate	Create a new Issuance Template with default configuration data.	Online
deleteIssuanceTemplate	Delete an existing Issuance Template.	Online Offline
Validation Template Commands		
getAllValidationTemplates	Retrieve the names of all the existing Validation Templates.	Online Offline
getValidationTemplate	Retrieve configuration data of a specific Validation Template.	Online Offline
createWSSValidationTemplate	Create a new WS Security Validation Template with default configuration data.	Online Offline
createWSTrustValidationTemplate	Create a new WS Trust Validation Template with default configuration data.	Online Offline
deleteValidationTemplate	Delete an existing Issuance Template.	Online Offline
configureOWSMAgentSTS	Modify configuration to allows MSAS/OWSM policies to work	Online

getPartner

Online command that retrieves the Partner entry and prints out the configuration for this partner.

Description

Retrieves the Partner entry and prints out the configuration for this partner.

Syntax

```
getPartner(partnerId)
```

Argument	Definition
partnerId	Specifies the partnerId: the ID of the partner.

Example

The following invocation retrieves the Partner entry and prints out the configuration for `customPartner`:

```
getPartner(partnerId="customPartner")
```

getAllRequesterPartners

Online command that retrieves Requester type partners.

Description

Retrieves Requester type partners.

Syntax

```
getAllRequesterPartners()
```

Example

The following invocation retrieves Requester type partners:

```
getAllRequesterPartners()
```

getAllRelyingPartyPartners

Online command that retrieves Relying Party partners.

Description

Retrieves the Relying Party partners.

Syntax

```
getAllRelyingPartyPartners()
```

Example

The following invocation retrieves Relying Party partners:

```
getAllRelyingPartyPartners()
```

getAllIssuingAuthorityPartners

Online command that retrieves Issuing Authority partners and prints out the result.

Description

Retrieves the Issuing Authority partners and prints out the result.

Syntax

```
getAllIssuingAuthorityPartners()
```

Example

The following invocation retrieves Issuing Authority partners and prints out the result:

```
getAllIssuingAuthorityPartners()
```

isPartnerPresent

Online command that queries the Security Token Service to determine whether or not the specified partner exists in the Partner store.

Description

Queries the Security Token Service to determine whether or not the specified partner exists in the Partner store, and prints out the result.

Syntax

```
isPartnerPresent(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

Example

The following invocation queries the Security Token Service to determine whether or not `customPartner` exists in the Partner store, and prints out the result:

```
isPartnerPresent(partnerId="customPartner")
```

createPartner

Online command that creates a new Partner entry.

Description

Creates a new Partner entry based on provided information. Displays a message indicating the result of the operation.

Syntax

```
createPartner(partnerId, partnerType, partnerProfileId, description,
bIsTrusted)
```

Argument	Definition
partnerId	Specifies the ID of the new partner to be created.
partnerType	Specifies the type of partner. Values can be one of the following: ■ STS_REQUESTER for Requester ■ STS_RELYING_PARTY for Relying Party ■ STS_ISSUING_AUTHORITY for Issuing Authority
partnerProfileId	Specifies the profile ID to be attached to this partner. It must reference an existing partner profile, and the type of the partner profile must be compliant with the type of the new partner entry.
description	Specifies the optional description of this new partner entry.
bIsTrusted	A value that indicates whether or not this new partner is trusted. Value can be either: ■ true for trusted ■ false if not trusted

Example

The following invocation creates STS_Requestor partner, `customPartner`, `custom-partnerprofile` with a description (`custom requester`), with a trust value of `true`, displays a message indicating the result of the operation:

```
createPartner(partnerId="customPartner", partnerType="STS_REQUESTER",
partnerProfileId="custom-partnerprofile", description="custom requester",
bIsTrusted="true")
```

updatePartner

Online command that updates an existing Partner entry.

Description

Updates an existing Partner entry based on the provided information. Displays a message indicating the result of the operation.

Syntax

```
updatePartner(partnerId, partnerProfileId, description, bIsTrusted)
```

Argument	Definition
partnerId	Specifies the ID of the new partner to be updated.
partnerProfileId	Specifies the partner profile ID. It must reference an existing partner profile, and the type of the partner profile must be compliant with the type of the new partner entry.
description	Specifies the optional description f this new partner entry.
bIsTrusted	A value that indicates whether or not this new partner is trusted. Value can be either: ■ true for trusted ■ false if not trusted

Example

The following invocation updates `customPartner` with a new profile ID, (`x509-wss-validtemp`), description (`custom requester with new profile id`), and a trust value of `false`. A message indicates the result of the operation:

```
updatePartner(partnerId="customPartner", partnerProfileId="x509-wss-validtemp",
description="custom requester with new profile id", bIsTrusted="false")
```

deletePartner

Online command that deletes a partner entry from the Security Token Service.

Description

Deletes an existing Partner entry referenced by the `partnerId` parameter from the Security Token Service, and prints out the result of the operation.

Syntax

```
deletePartner(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner to be deleted.

Example

The following invocation deletes the `customPartner` partner entry referenced by the partnerId parameter from the Security Token Service, and prints out the result of the operation:

```
deletePartner(partnerId="customPartner")
```

getPartnerUsernameTokenUsername

Online command that retrieves a partner's username value that will be used for UNT credentials partner validation or mapping operation.

Description

Retrieves a partner's username value that will be used for UNT credentials partner validation or mapping operation, and displays the value.

Syntax

```
getPartnerUsernameTokenUsername(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

Example

The following invocation retrieves the `customPartner` partner username value that will be used for UNT credentials partner validation or mapping operation, and displays the value:

```
getPartnerUsernameTokenUsername(partnerId="customPartner")
```

getPartnerUsernameTokenPassword

Online command that retrieves a partner's password value that will be used for UNT credentials partner validation or mapping operation.

Description

Retrieves a partner password value that will be used for UNT credentials partner validation or mapping operation, and displays the value.

Syntax

```
getPartnerUsernameTokenPassword(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

Example

The following invocation retrieves `customPartner` partner password value that will be used for UNT credentials partner validation or mapping operation, and displays the value:

```
getPartnerUsernameTokenPassword(partnerId="customPartner")
```

setPartnerUsernameTokenCredential

Online command that sets the username and password values of a partner entry, that will be used for UNT credentials partner validation or mapping operation.

Description

Sets the username and password values of a partner entry, that will be used for UNT credentials partner validation or mapping operation. Displays the result of the operation.

Syntax

```
setPartnerUsernameTokenCredential(partnerId, UTUsername, UTPassword)
```

Argument	Definition
partnerId	Specifies the ID of the partner.
UTUsername	Specifies the username value used for UNT credentials validation or mapping operations.
UTPassword	Specifies the username value used for UNT credentials validation or mapping operations.

Example

The following invocation sets the username and password values of the `customPartner` partner entry, and displays the result of the operation:

```
setPartnerUsernameTokenCredential(partnerId="customPartner", UTUsername="test",
UTPassword="password")
```

deletePartnerUsernameTokenCredential

Online command that removes the username and password values from a partner entry that are used for UNT credentials partner validation or mapping operation, and displays the result of the operation.

Description

Removes the username and password values from a partner entry that are used for UNT credentials partner validation or mapping operation, and displays the result of the operation.

Syntax

```
deletePartnerUsernameTokenCredential(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner to be deleted.

Example

The following invocation removes the username and password values from a partner entry that are used for UNT credentials partner validation or mapping operation, and displays the result of the operation:

```
deletePartnerUsernameTokenCredential(partnerId="customPartner")
```

getPartnerSigningCert

Online command that retrieves the Base64 encoded signing certificate for the partner referenced by the partnerId parameter, and displays its value, as a Base64 encoded string.

Description

Retrieves the Base64 encoded signing certificate for the partner referenced by the partnerId parameter, and displays its value, as a Base64 encoded string.

Syntax

```
getPartnerSigningCert(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

Example

The following invocation retrieves Base64 encoded signing certificate for the partner referenced by the partnerId parameter, and displays its value, as a Base64 encoded string:

```
getPartnerSigningCert(partnerId="customPartner")
```

getPartnerEncryptionCert

Online command that retrieves the Base64 encoded encryption certificate, and displays its value as a Base64 encoded string.

Description

Retrieves the Base64 encoded encryption certificate for the partner referenced by the partnerId parameter, and displays its value as a Base64 encoded string.

Syntax

```
getPartnerEncryptionCert(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

Example

The following invocation retrieves the Base64 encoded encryption certificate for the partner referenced by the partnerId parameter, and displays its value, as a Base64 encoded string:

```
getPartnerEncryptionCert(partnerId="customPartner")
```

setPartnerSigningCert

Online command that Uploads the provided certificate to the partner entry as the signing certificate. Displays the result of the operation.

Description

Uploads the provided certificate to the partner entry (referenced by the partnerId parameter) as the signing certificate. The supported formats of the certificate are DER and PEM. Displays the result of the operation.

Syntax

```
setPartnerSigningCert(partnerId, certFile)
```

Argument	Definition
partnerId	Specifies the ID of the partner.
certFile	Specifies the location of the certificate on the local file system. Supported formats of the certificate are DER and PEM.

Example

The following invocation uploads the provided certificate to the partner entry `customPartner` as the signing certificate. Displays the result of the operation:

```
setPartnerSigningCert(partnerId="customPartner", certFile="/temp/signing_cert")
```

setPartnerEncryptionCert

Online command that Uploads the provided certificate to the partner entry as the encryption certificate. Displays the result of the operation.

Description

Uploads the provided certificate to the partner entry (referenced by the partnerId parameter) as the encryption certificate. Displays the result of the operation.

Syntax

```
setPartnerEncryptionCert(partnerId, certFile)
```

Argument	Definition
partnerId	Specifies the ID of the partner.
certFile	Specifies the location of the certificate on the local filesystem. Supported formats of the certificate are DER and PEM.

Example

The following invocation uploads the provided certificate to the partner entry `customPartner` as the signing certificate. Displays the result of the operation:

```
setPartnerSigningCert(partnerId="customPartner", certFile="/temp/signing_cert")
```

deletePartnerSigningCert

Online command that removes the encryption certificate from the partner entry and displays the result of the operation.

Description

Removes the encryption certificate from the partner entry, referenced by the partnerId parameter, and displays the result of the operation.

Syntax

```
deletePartnerSigningCert(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

Example

The following invocation removes the encryption certificate from the partner entry, `customPartner`, and displays the result of the operation:

```
deletePartnerSigningCert(partnerId="customPartner")
```

deletePartnerEncryptionCert

Online command that removes the signing certificate from the partner entry and displays the result of the operation.

Description

Removes the signing certificate from the partner entry, referenced by the partnerId parameter, and displays the result of the operation.

Syntax

```
deletePartnerEncryptionCert(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the partner.

Example

The following invocation removes the signing certificate from the partner entry, `customPartner`, and displays the result of the operation:

```
deletePartnerEncryptionCert(partnerId="customPartner")
```

getPartnerAllIdentityAttributes

Online command that retrieves and displays all the identity mapping attributes used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner.

Description

Retrieves and displays all the identity mapping attributes used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner.

The identity mapping attributes only exist for partners of type Requester.

Syntax

```
getPartnerAllIdentityAttributes(partnerId)
```

Argument	Definition
partnerId	Specifies the ID of the Requester partner. Identity mapping attributes only exist for partners of type Requester

Example

The following invocation retrieves and displays all the identity mapping attributes used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner: `customPartner`.

```
getPartnerAllIdentityAttributes(partnerId="customPartner")
```

getPartnerIdentityAttribute

Online command that retrieves and displays identity mapping attributes used to map a token or to map binding data to a requester partner.

Description

Retrieves and displays an identity mapping attribute used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner.

The identity mapping attributes only exist for partners of type Requester.

Syntax

```
getPartnerIdentityAttribute(partnerId, identityAttributeName)
```

Argument	Definition
partnerId	Specifies the ID of the Requester partner.
IdentityAttributeName	Specifies the name of the identity mapping attribute to retrieve and display. For example: `httpbasicusername`.

Example

The following invocation retrieves and displays one `identityAttribute` and its value as specified by `identityAttributeName`.

```
getPartnerIdentityAttribute(partnerId="customPartner",
identityAttributeName="httpbasicusername")
```

setPartnerIdentityAttribute

Online command that sets the identity mapping attribute for the Requester partner.

Description

Set the identity mapping attribute specified by `identityAttributeName` for the partner of type requester specified by the partnerId parameter. These identity mapping attributes only exist for Requester partners. Displays the result of the operation.

Syntax

```
setPartnerIdentityAttribute(partnerId, identityAttributeName,
identityAttributeValue)
```

Argument	Definition
partnerId	Specifies the ID of the partner of type Requester.
identityAttributeName	Specifies the name of the identity mapping attribute to retrieve and display.
identityAttributeValue	Specifies the value of the identity mapping attribute to set.

Example

The following invocation sets the identity mapping attribute specified by `identityAttributeName` for the Requester partner of type requester specified by the partnerId parameter. Displays the result of the operation.

```
setPartnerIdentityAttribute(partnerId="customPartner",
identityAttributeName="httpbasicusername",identityAttributeValue="test")
```

deletePartnerIdentityAttribute

Online command that deletes the identity mapping attribute.

Description

Deletes the identity mapping attribute specified by `identityAttributeName`.

The identity mapping attributes used to map a token to a requester partner, or to map binding data (SSL Client certificate or HTTP Basic Username) to a requester partner, and they only exist for Requester partners.

Syntax

```
deletePartnerIdentityAttribute(partnerId, identityAttributeName)
```

Argument	Definition
partnerId	Specifies the ID of the partner.
identityAttributeName	Specifies the name of the identity mapping attribute to delete.

Example

The following invocation deletes the identity mapping attribute specified by `identityAttributeName` for Requester partner `customPartner`.

```
deletePartnerIdentityAttribute(partnerId="customPartner",
identityAttributeName="httpbasicusername")
```

getAllWSPrefixAndPartnerMappings

Online command that retrieves and displays all WS Prefixes to Relying Party Partner mappings.

Description

Retrieves and displays all WS Prefixes to Relying Party Partner mappings.

Syntax

```
getAllWSPrefixAndPartnerMappings()
```

Example

The following invocation retrieves and displays theWS Prefixes.

```
getAllWSPrefixAndPartnerMappings()
```

getWSPrefixAndPartnerMapping

Online command that retrieves and displays the Relying Party Partner mapped to the specified wsprefix parameter, if a mapping for that WS Prefix exists.

Description

Retrieves and displays the Relying Party Partner mapped to the specified wsprefix parameter, if a mapping for that WS Prefix exists.

Syntax

```
getWSPrefixAndPartnerMapping(wsprefix)
```

Argument	Definition
wsprefix	Specifies the WS Prefix entry to retrieve and display. The path is optional. If specified, it should take the following form: *http_protocol://hostname_ip/path*

Example

The following invocation retrieves nd displays the Relying Party Partner mapped to the specified wsprefix parameter, if a mapping for that WS Prefix exists.

```
getWSPrefixAndPartnerMapping(wsprefix="http://host1.example.com/path")
```

createWSPrefixAndPartnerMapping

Online command that creates a new WS Prefix mapping to a Relying Partner.

Description

Creates a new WS Prefix mapping to a Relying Partner referenced by the partnerid parameter, and displays the result of the operation.

Syntax

```
createWSPrefixAndPartnerMapping(wsprefix, partnerid, description)
```

Argument	Definition
wsprefix	Specifies the WS Prefix entry to retrieve and display. The path is optional. If specified, it should take the following form: *http_protocol://hostname_ip/path*
partnerId	Specifies the ID of the partner.
description	Specifies an optional description.

Example

The following invocation creates a new WS Prefix mapping to a Relying Partner Partner referenced by the partnerid parameter, and displays the result of the operation.

```
createWSPrefixAndPartnerMapping(wsprefix="http://host1.example.com/path",
partnerid="customRPpartner", description="some description")
```

deleteWSPrefixAndPartnerMapping

Online command that deletes an existing mapping of WS Prefix to a Relying Partner Partner.

Description

Deletes an existing mapping of WS Prefix to a Relying Partner, and displays the result of the operation.

Syntax

```
deleteWSPrefixAndPartnerMapping(wsprefix)
```

Argument	Definition
wsprefix	Specifies the WS Prefix entry to retrieve and display. The path is optional. If specified, it should take the following form: *http_protocol://hostname_ip/path*

Example

The following invocation deletes the existing mapping of WS Prefix to a Relying Partner, and displays the result of the operation.

```
deleteWSPrefixAndPartnerMapping(wsprefix="http://host1.example.com/path")
```

getAllPartnerProfiles

Online command that retrieves the names of all the existing partner profiles and displays them.

Description

Retrieves the names of all the existing partner profiles and displays them.

Syntax

```
getAllPartnerProfiles()
```

Example

The following invocation retrieves the names of all the existing partner profiles and displays them.

```
getAllPartnerProfiles()
```

getPartnerProfile

Online command that retrieves the configuration data of a specific partner profile, and displays the content of the profile.

Description

Retrieves the configuration data of the partner profile referenced by the partnerProfileId parameter, and displays the content of the profile.

Syntax

```
getPartnerProfile(partnerProfileId)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile.

Example

The following invocation retrieves the configuration data of the partner profile referenced by the partnerProfileId parameter, and displays the content of the profile.

```
getPartnerProfile(partnerProfileId="custom-partnerprofile")
```

createRequesterPartnerProfile

Online command that creates a new requester partner profile with default configuration data.

Description

Creates a new requester partner profile with default configuration data, and displays the result of the operation.

Table 8–2 describes the default configuration created with this command.

Table 8–2 Default Configuration: createRequesterPartnerProfile

Element	Description
Return Error for Missing Claims	Default: false
Allow Unmapped Claims	Default: false

Token Type Configuration	The Token Type Configuration table includes the following entries. There are no mappings of token type to WS-Trust Validation Template:
	■ SAML 1.1 token type mapped to the following External URI:
	`http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1`
	The SAML 1.1 token type is not mapped to any WS-Trust Validation Template.
	■ SAML 2.0 token type mapped to the following External URI:
	`http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0`
	The SAML 2.0 token type is not mapped to any WS-Trust Validation Template.
	■ Username token type mapped to the following External URI:
	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken`
	The Username token type is not mapped to any WS-Trust Validation Template.
	Note: Token Type Configuration and token type to Validation Template mapping are both empty

Attribute Name Mapping	Default: The Attribute Name Mapping table is empty by default.

Syntax

```
createRequesterPartnerProfile(partnerProfileId, defaultRelyingPartyPPID,
description)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile.
defaultRelyingPartyPPID	Specifies the relying party partner profile to use, if the AppliesTo field is missing from the RST or if it could not be mapped to a Relying Party Partner.
description	Specifies the optional description for this partner profile

Example

The following invocation creates a new requester partner profile with default configuration data, and displays the result of the operation. For default data descriptions, see Table 8–2.

```
createRequesterPartnerProfile(partnerProfileId="custom-partnerprofile",
defaultRelyingPartyPPID="rpPartnerProfileTest", description="custom
partner profile")
```

createRelyingPartyPartnerProfile

Online command that creates a new relying party partner profile with default configuration data.

Description

Creates a new relying party partner profile with default configuration data, and displays the result of the operation.

Table 8–3 describes the default configuration created with this command.

Table 8–3 Default Configuration: createRelyingPartyPartnerProfile

Element	Description
Download Policy	Default: false
Allow Unmapped Claims	Default: false
Token Type Configuration	The Token Type Configuration will contain a single entry, with: ▪ The token type set to the type of Issuance Template referenced by defaultIssuanceTemplateID ▪ The Issuance template set to defaultIssuanceTemplateID **Note**: For the token type of the issuance template referenced by defaultIssuanceTemplateID, it will be linked to the issuance template, while the other token types will not be linked to any issuance template. If the issuance template referenced by defaultIssuanceTemplateID is of custom token type, the table will only contain one entry, with the custom token type, mapped to the custom token type as the external URI, and mapped to the issuance template referenced by defaultIssuanceTemplateID
Attribute Name Mapping	The Attribute Name Mapping table is empty be default.

Syntax

```
createRelyingPartyPartnerProfile(partnerProfileId, defaultIssuanceTemplateID,
description)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile.
defaultIssuanceTemplateID	Specifies the default issuance template and token type to issue if no token type was specified in the RST.
description	Specifies the optional description for this partner profile

Example

The following invocation creates a new relying party partner profile with default configuration data, and displays the result of the operation.

```
createRelyingPartyPartnerProfile(partnerProfileId="custom-partnerprofile",
defaultIssuanceTemplateID="saml11-issuance-template", description="custom partner
profile")
```

createIssuingAuthorityPartnerProfile

Online command that creates a new issuing authority partner profile with default configuration data.

Description

Creates a new issuing authority partner profile with the default configuration data in Table 8–4, and displays the result of the operation.

Table 8–4 Default Configuration: createIssuingAuthorityPartnerProfile

Element	Description
Server Clockdrift	Default: 600 seconds
Token Mapping	The Token Mapping Section will be configured as follows: ■ Override Simple User Mapping: false ■ Override User NameID Mapping: false ■ Override Attribute Based User Mapping: false ■ Override Simple Partner Mapping: false ■ Override Partner NameID Mapping: false Empty fields ■ simple user mapping ■ attribute based user mapping ■ simple partner mapping
Partner NameID Mapping	The Partner NameID Mapping table will be provisioned with the following entries as NameID format. However, without any data in the datastore column the issuance template referenced by defaultIssuanceTemplateID is of token type SAML 1.1, SAML 2.0, or Username. The table will contain the following entries: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

Table 8–4 (Cont.) Default Configuration: createIssuingAuthorityPartnerProfile

Element	Description
User NameID Mapping	The User NameID Mapping table will be provisioned with the following entries as NameID format: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName, empty datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName, dn set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress, mail set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent, empty datastore column
Attribute Mapping	The Attribute Value Mapping and Attribute Name Mapping table is empty be default.

Syntax

```
createIssuingAuthorityPartnerProfile(partnerProfileId, description)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile.
description	Specifies the optional description for this partner profile

Example

The following invocation a new issuing authority partner profile with default configuration data, and displays the result of the operation.

```
createIssuingAuthorityPartnerProfile(partnerProfileId="custom-partnerprofile"
description="custom partner profile")
```

deletePartnerProfile

Online command that deletes an partner profile referenced by the partnerProfileId parameter. (See Section 5.2, "Advanced Identity Federation Commands" for information regarding SAML 1.1.)

Description

Deletes an partner profile referenced by the partnerProfileId parameter, and displays the result of the operation.

Syntax

```
deletePartnerProfile(partnerProfileId)
```

Argument	Definition
partnerProfileId	Specifies the name of the partner profile to be removed.

Example

The following invocation deletes an partner profile referenced by the partnerProfileId parameter, and displays the result of the operation.

```
deletePartnerProfile(partnerProfileId="custom-partnerprofile")
```

getAllIssuanceTemplates

Online command that retrieves the names of all the existing issuance templates.

Description

Retrieves the names of all the existing issuance templates and displays them.

Syntax

```
getAllIssuanceTemplates
```

Example

The following invocation retrieves the names of all the existing issuance templates and displays them.

```
getAllIssuanceTemplates
```

getIssuanceTemplate

Online command that retrieves the configuration data of a specific issuance template.

Description

Retrieves the configuration data of the issuance template referenced by the issuanceTemplateId parameter, and displays the content of the template.

Syntax

```
getIssuanceTemplate(issuanceTemplateId)
```

Argument	Definition
issuanceTemplateId	Specifies the name of the issuance template.

Example

The following invocation retrieves the configuration data of the issuance template referenced by the issuanceTemplateId parameter, and displays the content of the template.

```
getIssuanceTemplate(issuanceTemplateId="custom-issuancetemp")
```

createIssuanceTemplate

Online command that creates a new issuance template with default configuration data.

Description

Creates a new issuance template with default configuration data, and displays the result of the operation.

Table 8–5 describes the default configuration for this command.

Table 8–5 Default Configuration: createIssuanceTemplate

Token Type	Description
Username	The issuance template will be created with the following default values: ■ Send Encrypted Token: false ■ NameID User Attribute: uid ■ NameID User Attribute Store: User Store ■ Password Attribute: (empty) ■ Include Nonce: true ■ Include Timestamp: true
SAML 1.1 or SAML 2.0	The issuance template will be created with the following default values: ■ Send Encrypted Token: false ■ Assertion Issuer: Access Manager Hostname ■ NameID Format: Email Address ■ NameID User Attribute: mail ■ NameID User Attribute Store: User Store ■ NameID Qualifier: (empty) ■ Include Authn Statement: true ■ Include Attr Statement: true ■ Sign Assertion: true ■ Include Certificate in Signature: true ■ Send Encrypted NameID: false (SAML 2.0 only) ■ Default Subject Confirmation Method: Sender Vouches ■ Compute HOK Symmetric Key: true ■ HOK Symmetric Key Generation Algorithm: `http://www.w3.org/2001/04/xmlenc#aes128-cbc` Empty tables: Attribute Name Mapping, Attribute Value Mapping and Attribute Value Filter
Custom Type	The issuance template will be created with the following default values: ■ Send Encrypted Token: false

Syntax

```
createIssuanceTemplate(issuanceTemplateId, tokenType, signingKeyId,
description)
```

Argument	Definition
issuanceTemplateId	Specifies the name of the issuance template to be created.
tokenType	Possible values can be: ■ username: indicates that the token type is UsernameToken ■ saml11: indicates that the token type is a SAML 1.1 Assertion ■ saml20: indicates that the token type is a SAML 2.0 Assertion ■ <other>: in this case, the token type is assumed to be a custom token type, referenced by <other> (replace <other> by a value)
signingKeyId	Specifies the keyID referencing the key entry (defined in the STS General Settings UI section) that will be used to sign outgoing SAML Assertions. Only required when token type is saml11 or saml20.
description	An optional description.

Example

The following invocation creates a new issuance template with default configuration data, and displays the result of the operation.

```
createIssuanceTemplate(issuanceTemplateId="custom-issuancetemp",
tokenType="saml20", signingKeyId="osts_signing", description="custom issuance
template")
```

deleteIssuanceTemplate

Online command that deletes an issuance template referenced by the issuanceTemplateId parameter, and displays the result of the operation.

Description

Deletes an issuance template referenced by the issuanceTemplateId parameter, and displays the result of the operation.

Syntax

```
deleteIssuanceTemplate(issuanceTemplateId)
```

Argument	Definition
issuanceTemplateId	Specifies the name of the existing issuance template to be removed.

Example

The following invocation deletes an issuance template referenced by the issuanceTemplateId parameter, and displays the result of the operation.

```
deleteIssuanceTemplate(issuanceTemplateId="custom-issuancetemp")
```

getAllValidationTemplates

Online command that retrieves the names of all the existing validation templates.

Description

Retrieves the names of all the existing validation templates and displays them.

Syntax

```
getAllValidationTemplates()
```

Example

The following invocation retrieves the names of all the existing validation templates and displays them.

```
getAllValidationTemplates()
```

getValidationTemplate

Online command that retrieves the configuration data of a specific validation template, and displays the content of the template.

Description

Retrieves the configuration data of the validation template referenced by the validationTemplateId parameter, and displays the content of the template.

Syntax

```
getValidationTemplate(validationTemplateId)
```

Argument	Definition
validationTemplateId	Specifies the name of the existing validation template.

Example

The following invocation retrieves the configuration data of a specific validation template, and displays the content of the template.

```
getValidationTemplate(validationTemplateId="custom-wss-validtemp")
```

createWSSValidationTemplate

Online command that creates a new validation template with default configuration data.

Description

Creates a new WSS validation template with default configuration data, and displays the result of the operation. The validation template is created using the values in Table 8–6, depending on the token type.

Table 8–6 Default Configuration: createWSSValidationTemplate

Token Type	Description
Username	The validation template will be created with the following default values: ■ Timestamp Lifespan: 600 seconds ■ Enable Credential Validation: true ■ Validation Source: Partner ■ Token Mapping: Map token to Partner ■ Enable Simple Partner Mapping: true ■ Partner Datastore Attribute: username

Table 8–6 (Cont.) Default Configuration: createWSSValidationTemplate

Token Type	Description
SAML 1.1 or SAML 2.0	The validation template will be created with the following default values: ■ Authentication Timeout: 3600 seconds ■ Timestamp Lifespan: 3600 seconds The Token Mapping section will be created with the following default values: ■ Map token: Map token to Partner ■ Enable Simple User Mapping: false ■ Enable User NameID Mapping: false ■ Enable Attribute Based User Mapping: false ■ Enable Simple Partner Mapping: false ■ Enable Partner NameID Mapping: false Empty fields: User Token Attribute, User Datastore Attribute and Attribute Based User Mapping Also: ■ Partner Token Attribute: NameID ■ Partner Datastore Attribute: username Partner NameID Mapping table will be provisioned with the following entries as NameID format, but without any data in the datastore column: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualified Name ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent User NameID Mapping table will be provisioned with the following entries as NameID format: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualified Name, empty datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName, dn set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress, mail set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent, empty datastore column

Table 8–6 (Cont.) Default Configuration: createWSSValidationTemplate

Token Type	Description
X.509	The Token Mapping section will be created with the following default values: ■ Map token: Map token to Partner ■ Enable Simple User Mapping: false ■ Enable Attribute Based User Mapping: false ■ Enable Simple Partner Mapping: true Empty fields: User Token Attribute, User Datastore Attribute and Attribute Based User Mapping Also: ■ Partner Token Attribute: DN ■ Partner Datastore Attribute: sslclientcertdn
Kerberos	The Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: true ■ Enable Attribute Based User Mapping: false ■ Enable Simple Partner Mapping: false Empty fields: Partner Token Attribute, Partner Datastore Attribute and Attribute Based User Mapping Also: ■ User Token Attribute: TPE_KERBEROS_PRINCIPAL_FULL ■ User Datastore Attribute: mail

Syntax

```
createWSSValidationTemplate(templateId, tokenType,
defaultRequesterPPID, description)
```

Argument	Definition
templateId	Specifies the name of the name of the validation template to be created.
tokenType	Specifies the token type of the validation template. Possible values can be: ■ username: indicates that the token type is UsernameToken ■ saml11: indicates that the token type is a SAML 1.1 Assertion ■ saml20: indicates that the token type is a SAML 2.0 Assertion ■ x509: indicates that the token type is an X.509 certificate ■ kerberos: indicates that the token type is a Kerberos token ■ oam: indicates that the token type is Access Manager
defaultRequesterPPID	Specifies the Requester partner profile to use if OSTS is configured not to map the incoming message to a requester.
description	Specifies an optional description.

Example

The following invocation creates a new validation template with default configuration data, and displays the result of the operation.

```
createWSSValidationTemplate(templateId="custom-wss-validtemp", tokenType="custom",
defaultRequesterPPID="requesterPartnerProfileTest", description="custom validation
template")
```

createWSTrustValidationTemplate

Online command that creates a new WS-Trust validation template with default configuration data.

Description

Creates a new WS-Trust validation template with default configuration data, and displays the result of the operation. The WS-Trust validation template is created with the values in Table 8–7, depending on the token type.

Table 8–7 Default Configuration: createWSTrustValidationTemplate

Token Type	Description
Username	The WS-Trust validation template will be created with the following default values: ■ Timestamp Lifespan: 600 seconds ■ Enable Credential Validation: false ■ Validation Source: User Store ■ Token Mapping: Map token to User ■ Enable Simple User Mapping: true ■ USer Datastore Attribute: uid
SAML 1.1 or SAML 2.0	The WS-Trust validation template will be created with the following default values: ■ Authentication Timeout: 3600 seconds ■ Timestamp Lifespan: 3600 seconds The Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: false ■ Enable User NameID Mapping: true ■ Enable Attribute Based User Mapping: false Empty fields: User Datastore Attribute, Attribute Based User Mapping User NameID Mapping table will be provisioned with the following entries as NameID format: ■ urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName, empty datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName, dn set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress, mail set in the datastore column ■ urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos, empty datastore column ■ urn:oasis:names:tc:SAML:2.0:nameid-format:persistent, empty datastore column

Table 8–7 (Cont.) Default Configuration: createWSTrustValidationTemplate

Token Type	Description
X.509	The WS-Trust Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: true ■ Enable Attribute Based User Mapping: false ■ Enable Simple Partner Mapping: true ■ User Token Attribute: CN ■ User Datastore Attribute: CN ■ Attribute Based User Mapping (empty)
Kerberos	The WS-Trust Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: true ■ Enable Attribute Based User Mapping: false ■ Attribute Based User Mapping (empty) ■ User Token Attribute: TPE_KERBEROS_PRINCIPAL_FULL ■ User Datastore Attribute: mail
OAM	The WS-Trust Token Mapping section will be created with the following default values: ■ Map token: Map token to User ■ Enable Simple User Mapping: true ■ Enable Attribute Based User Mapping: false ■ Attribute Based User Mapping (empty) ■ User Token Attribute: TPE_NAME_ID ■ User Datastore Attribute: uid
custom	The WS-Trust Token Mapping section will be created with the following default values: ■ Map token: Map token to None ■ Enable Simple User Mapping: false ■ Enable Attribute Based User Mapping: false ■ Attribute Based User Mapping (empty) ■ User Token Attribute: (empty) ■ User Datastore Attribute: (empty)

Syntax

```
createWSTrustValidationTemplate(templateId, tokenType, description)
```

Argument	Definition
templateId	Specifies the name of the name of the WS-Trust validation template to be created.

Argument	Definition
tokenType	Specifies the token type of the WS-Trust validation template. Possible values can be: ■ username: indicates that the token type is UsernameToken ■ saml11: indicates that the token type is a SAML 1.1 Assertion ■ saml20: indicates that the token type is a SAML 2.0 Assertion ■ x509: indicates that the token type is an X.509 certificate ■ kerberos: indicates that the token type is a Kerberos token ■ oam: indicates that the token type is an Access Manager token, supported by default ■ <other>: in this case, the token type is assumed to be a custom token type, referenced by <other> (replace <other> by a value)
description	Specifies an optional description.

Example

The following invocation creates a new WS-Trust validation template with default configuration data, and displays the result of the operation.

```
createWSTrustValidationTemplate(templateId="custom-wss-validtemp",
tokenType="custom", description="custom validation template")
```

deleteValidationTemplate

Online command that deletes a validation template.

Description

Deletes a validation template referenced by the validationTemplateId parameter, and displays the result of the operation.

Syntax

```
deleteValidationTemplate(validationTemplateId)
```

Argument	Definition
validationTemplateId	Specifies the name of the validation template to be removed.

Example

The following invocation deletes a validation template referenced by the validationTemplateId parameter, and displays the result of the operation.

```
deleteValidationTemplate(validationTemplateId="custom-wss-validtemp")
```

configureOWSMAgentSTS

Online command required to allow custom Mobile Security Access Server (MSAS)/Oracle Web Services Manager (OWSM) policies to work.

Description

Online command modifies the Security Token Service configuration to allow custom MSAS/OWSM policies to work.

Syntax

```
configureOWSMAgentSTS(<type>, <server="soa_server1">)
```

Argument	Definition
type	The type can be 'classpath' or 'policymanager'. If OWSM policy manager service has to be seeded with STS policy, then pass in 'policymanager'. Otherwise, use 'password' and STS policies are picked from sts_policies.jar.
server	Optional. If type=policymanager, enter the WLS managed server name where the OWSM Document Manager MBean is deployed.

Example

```
configureOWSMAgentSTS("policymanager", server="omsm_server1")
```

Part II

WLST for Oracle Mobile Security Suite

Part II describes the WLST commands for Mobile Access Security Server.

9

Mobile Security Access Server WLST Commands

This section provides detailed descriptions of custom WebLogic Scripting Tool (WLST) commands for Mobile Access Security Server (MSAS), including command syntax, arguments and command examples. These commands perform many of the same functions that you can complete using MSAS console.

The following topics describe the WLST commands for managing MSAS instances from the command line, including configuring identity store profiles, message security, and trusted issuers. Topics include:

- Using the WLST Commands
- MSAS Configuration Commands
- MSAS Identity Store Profile Commands
- Repository Commands
- Session Commands
- Token Issuer Trust Configuration Commands
- Diagnostic Commands

Notes:

- This chapter describes only the WLST commands supported by MSAS.
- To learn more about configuring an MSAS using WLST, see "Configuring an MSAS Instance Using WLST" in *Administering Mobile Security Access Server*.

9.1 Using the WLST Commands

The WLST commands for managing MSAS must be executed from the *IDM_HOME*`/common/bin directory` directory of your Mobile Security Manager (MSM), for example `/home/oracle/omsm/ORACLE_IDM/common/bin`. Before running these commands, ensure that the Administration Server for the MSM domain is running.

To display the help for the MSAS configuration, identity store, and repository management commands, connect to a running instance of the server and enter `help('msasManage')`.

To display the help for `resetWSMPolicyRepository`, session commands, and trusted issuer configuration commands, connect to a running instance of the server and enter `help('wsmManage')`.

9.2 MSAS Configuration Commands

Use the commands listed in Table 9–1 to manage MSAS configurations and application metadata.

> **Note:** The MSAS configuration command definitions begin with "displayMSASConfiguration" on page 9-6.

Table 9–1 MSAS Configuration Commands

Use this command...	To...	Use with WLST...
displayMSASConfiguration	Display the MSAS configuration properties, and their values and groups, for a MSAS instance.	Online
setMSASConfiguration	Create or modify the configuration properties for a MSAS instance.	Online
setMSASLogLevel	Set the logger configuration for a MSAS instance to a specified level.	Online
getMSASLogLevel	Get logger configuration for a MSAS instance.	Online
listMSASLoggers	List the logger configurations for a MSAS instance.	Online

9.3 MSAS Identity Store Profile Commands

Use these commands to manage an identity store profile, which is a logical representation of a user repository. The identity store configuration is stored in an Identity Profile Document in the MSAS Repository.

All identity store profile management commands must be performed in the context of a repository session. A repository session can only act on a single document.

Use the commands listed in Table 9–2 to manage MSAS identity store profiles.

> **Note:** The MSAS configuration command definitions begin with "createIdentityProfile" on page 9-11.

Table 9–2 Identity Profile Management Commands

Use this command...	To...	Use with WLST...
createIdentityProfile	Create a new identity store profile in a MSAS instance within a repository session.	Online
displayIdentityProfile	Display the contents of a specified identity store profile, or list the names of all identity profiles, in a MSAS instance.	Online

Table 9–2 (Cont.) Identity Profile Management Commands

Use this command...	To...	Use with WLST...
selectIdentityProfile	Select an identity store profile in a MSAS instance for modification within a repository session.	Online
deleteIdentityProfile	Delete an identity store profile from a specified MSAS instance within a repository session.	Online
setIdentityProfileDirectory	Set or update the directory information for an identity store profile in a MSAS instance within a repository session.	Online
setIdentityProfileUser	Set or update the user information for an identity store profile in a MSAS instance within a repository session.	Online
setIdentityProfileGroup	Set or update the group information for an identity store profile for a MSAS instance within a repository session.	Online

9.4 Repository Commands

Use the commands listed in Table 9–3 to manage the documents stored in the Oracle Repository. For additional information about upgrading or migrating documents in an Oracle Repository, see "Migrating Applications in the Repository" in *Administering Mobile Security Access Server*.

Note: The MSAS configuration command definitions begin with "exportMSASAppMetadata" on page 9-19.

Table 9–3 Policy Repository Management Commands

Use this command...	To...	Use with WLST...
exportMSASAppMetadata	Export MSAS application metadata from the repository to a specified ZIP archive.	Online
importMSASAppMetadata	Import MSAS application metadata into the repository from a specified ZIP archive.	Online
migrateMSASAppHostports	Migrate physical host:port values for applications in a MSAS instance to a mapped host:port value.	Online
resetWSMPolicyRepository	Delete the existing policies stored in the repository and refresh it with the latest set of predefined policies that are provided in the new installation of the Oracle MSAS software.	Online

9.5 Session Commands

Some MSAS WLST commands, such as those that modify repository documents and trusted token issuers need to be executed in the context of a session. Use the WLST commands listed in Table 9–4 to manage a session.

Note: The MSAS configuration command definitions begin with "abortRepositorySession" on page 9-25.

Table 9–4 Session Management WLST Commands

Use this command...	To...	Use with WLST...
abortRepositorySession	Abort the current modification session, discarding any changes that were made during the session.	Online
beginRepositorySession	Begin a session to modify the repository documents.	Online
commitRepositorySession	Write the contents of the current session to the repository.	Online
describeRepositorySession	Describe the contents of the current session. This will indicate either that the session is empty or list the name of the document that is being updated, along with the type of update (create, modify, or delete).	Online

9.6 Token Issuer Trust Configuration Commands

Use the WLST commands listed in Table 9–5 to view and define trusted issuers, trusted distinguished name (DN) lists, and token attribute rule filters for trusted SAML or JWT signing certificates.

When using WLST to create, modify, and delete token issuer trust documents, you must execute the commands in the context of a session. Each session applies to a single trust document only.

Note: To view the help for the WLST commands described in this section, connect to a running instance of the server and enter `help('wsmManage')`.

For additional information about using these commands, see "Configuring Trusted Issuers and DN Lists Using WLST" in *Administering Mobile Security Access Server*.

Note: The MSAS configuration command definitions begin with "abortRepositorySession" on page 9-25.

Table 9–5 Token Issuer Trust Commands

Use this command...	To...	Use with WLST...
createWSMTokenIssuerTrustDocument	Create a new token issuer trust document using the name provided.	Online
deleteWSMTokenIssuerTrust	Delete the entry for the issuer, including the DN list in it.	Online
deleteWSMTokenIssuerTrustAttributeRule	Delete a token attribute rule associated with a trusted DN.	Online

Table 9–5 (Cont.) Token Issuer Trust Commands

Use this command...	To...	Use with WLST...
deleteWSMTokenIssuerTrustDocument	Delete the token issuer trust document, specified by the name argument, from the repository.	Online
displayWSMTokenIssuerTrust	Display the names of the DN lists associated with a specified issuer.	Online
exportWSMTokenIssuerTrustMetadata	Export the trust configuration (issuers, DNs, and token attribute rules) for all trusted issuers.	Online
importWSMTokenIssuerTrustMetadata	Import the trust configuration (Issuers, DNs, and token attribute rules) for all trusted issuers.	Online
listWSMTokenIssuerTrustDocuments	List the token issuer trust documents in the repository.	Online
revokeWSMTokenIssuerTrust	Remove trusted issuers and associated configurations (DNs and token attribute rules).	Online
selectWSMTokenIssuerTrustDocument	Select the token issuer trust document, identified by the name argument, to be modified in the session.	Online
setWSMTokenIssuerTrust	Specify a trusted issuer with a DN list.	Online
setWSMTokenIssuerTrustAttributeFilter	Add, delete, or update token attribute rules for a given token signing certificate DN.	Online
setWSMTokenIssuerTrustAttributeMapping	Specify the DN of a token signing certificate and a list of trusted users. The name ID and the attribute can be mapped to another user ID.	Online
setWSMTokenIssuerTrustDisplayName	Sets or resets the display name of the Token Issuer Trust document currently selected in the session.	Online

9.7 Diagnostic Commands

Use the WLST diagnostic command to check the status of the Oracle components that are required for proper functioning of the product. The MSAS configuration command definition is "checkWSMStatus" on page 9-45.

displayMSASConfiguration

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Displays the configuration properties, and their values and groups, for a MSAS instance. If a property is not defined in the MSAS instance's configuration document, then the default value defined for the product is displayed. If a MSAS instance name is not specified, or the specified MSAS instance does not exist, then the default values defined for the product are displayed.

Syntax

```
displayMSASConfiguration(instanceName=None)
```

Argument	Definition
instanceName	Optional. The name of the MSAS instance associated with the configuration document from which property values are displayed.

Example

This example displays the MSAS configuration properties, including their values and groups, for an instance named `MSAS-123456`.

```
wls:/mydomain/serverConfig> displayMSASConfiguration('MSAS-123456')
```

This example displays the default MSAS configuration properties.

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

setMSASConfiguration

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Sets the configuration properties in the configuration document associated with a MSAS instance. If the configuration document does not exist for the instance, it is created automatically. A new property with values and/or groups of values can be added inside the configuration document. The set of acceptable properties are determined by the default set of properties supported by the product. Specific property values or groups of values can also be removed from the configuration document. The configuration document is removed if it does not have any properties.

For additional information about using the `setMSASConfiguration` command, see "Configuring an MSAS Instance Using WLST" in *Administering Mobile Security Access Server*.

Syntax

```
setMSASConfiguration(instanceName,categoryName,propertyName,group=None,values=None
)
```

Argument	Definition
instanceName	The name of the MSAS instance associated with the configuration document to be modified.
categoryName	The category of the configuration property. This is verified with the default set of properties for acceptability.
propertyName	The name of the configuration property. This is verified with the default set of properties for acceptability.
groupName	Optional. The group containing the set of values to add in the configuration document. If the group exists, and this value is set to `None`, then the group is removed.
propertyValues	Optional. The array of values to set for a property or group inside the configuration document. Default is None which refers to an empty array list

Example

This example is setting the location of the SSL keystore in a MSAS instance named `MSAS-123456`.

```
wls:/mydomain/serverConfig>
setMSASConfiguration('MSAS-123456','ServerSettings','ssl.keystore.location',None,[
'kss://mag1/sslkeystore'])
```

setMSASLogLevel

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Sets the logger configuration for a MSAS instance to a specified level (for example, `WARNING`).

Syntax

```
setMSASLogLevel(instanceName,logger,level)
```

Argument	Definition
instanceName	The name of the MSAS instance logging configuration to be modified. This option is required; there is no default value.
logger	A logger name. The empty string denotes the root logger. This option is required; there is no default value.
level	The level name. It has to be one of the following: ■ SEVERE ■ WARNING ■ INFO ■ CONFIG ■ FINE ■ FINER ■ FINEST An empty string can be used to set the level to null (inherited from parent). This option is required; there is no default value.

Example

This example is setting a logging level of `FINEST` for an MSAS logger named `oracle.idm.gateway.grs` in an instance named `MSAS-123456`.

```
wls:/mydomain/serverConfig>
setMSASLogLevel('MSAS-123456','oracle.idm.gateway.grs','FINEST'])
```

This example is setting a logging level to null for an MSAS logger named `oracle.idm.gateway.grs` in an instance named `MSAS-123456`.

```
wls:/mydomain/serverConfig>
setMSASLogLevel('MSAS-123456','oracle.idm.gateway.grs',''])
```

This example is setting a logging level of `SEVERE` for the root logger in an instance named `MSAS-123456`.

```
wls:/mydomain/serverConfig> setMSASLogLevel('MSAS-123456','','SEVERE'])
```

getMSASLogLevel

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Gets the logger configuration for a MSAS instance.

Syntax

```
getMSASLogLevel(instanceName,logger)
```

Argument	Definition
instanceName	The MSAS instance name of the logging configuration. This option is required; there is no default value.
logger	A logger name. The empty string denotes the root logger. This option is required; there is no default value.

Example

This example is getting the logging level for an MSAS logger named `oracle.idm.gateway.grs` in an instance named `MSAS-123456`.

```
wls:/mydomain/serverConfig>
getMSASLogLevel('MSAS-123456','oracle.idm.gateway.grs')
```

This example is getting the logging level for the root MSAS logger in an instance named `MSAS-123456`.

```
wls:/mydomain/serverConfig>
getMSASLogLevel('MSAS-123456','oracle.idm.gateway.grs')
```

listMSASLoggers

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Lists the logger configurations for a MSAS instance.

Syntax

```
listMSASLoggers(instanceName)
```

Argument	Definition
instanceName	The MSAS instance name of the logging configuration. This option is required; there is no default value.

Example

This example is listing the loggers for a MSAS instance named `MSAS-123456`.

```
wls:/mydomain/serverConfig> listMSASLoggers('MSAS-123456'])
```

createIdentityProfile

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Create a new identity store profile in a MSAS instance within a repository session. The identity store profile must be associated with a specified MSAS instance.

Issuing this command outside of a repository session will result in an error.

Notes:

- This command only creates a new identity store profile for a MSAS instance. To use an identity store profile at runtime, the MSAS configuration must already be set, as described in setMSASConfiguration. After running `setMSASConfiguration`, you must also restart the Oracle MSAS Management Server for the runtime to access the configured profile. For more information about restarting the management server, see "Managing MSAS Instances" in *Administering Mobile Security Access Server*.
- You cannot create multiple identity store profiles in the same session. You must commit the session after creating a profile, and then start a new session to create another profile.
- If an error occurs during the creation of an identity store profile, the profile must first be committed before it can be deleted.

Syntax

```
createIdentityProfile(instanceName,Profilename,description=None)
```

Argument	Definition
instanceName	Name of the MSAS instance in which you want to create an identity store profile.
ProfileName	Name of the new identity store profile.
description	Optional. Description of the new identity store profile.

Example

This example creates an identity store profile named `identity-profile` in a MSAS instance named `MSAS-123456`, with a description of `Identity profile`.

```
wls:/mydomain/serverConfig>
createIdentityProfile('MSAS-123456','identity-profile','Identity profile')
```

displayIdentityProfile

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Display the contents of a specified identity store profile in a MSAS instance, or list the names of all identity store profiles associated with a specified MSAS instance.

If this command is executed from an active session, session changes are also displayed. If this command is executed outside of active session, content from the repository is displayed.

Syntax

```
displayIdentityProfile(instanceName,ProfileName=None)
```

Argument	Definition
instance	Name of the MSAS instance associated with the identity store profiles to be displayed.
ProfileName	Optional. The name of the identity store profile to be displayed. If a name is not specified, then the names of the all available identity profiles for the MSAS instance are displayed.

Example

This example displays the contents of an identity store profile named `identity-profile` in an instance named `MSAS-123456`.

```
wls:/mydomain/serverConfig>
displayIdentityProfile('MSAS-123456','identity-profile')
```

This example displays the names of the all available identity store profiles in the MSAS instance `MSAS-123456`.

```
wls:/mydomain/serverConfig> displayIdentityProfile('MSAS-123456')
```

selectIdentityProfile

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Select an identity store profile for modification in a MSAS instance within a repository session.

Syntax

```
selectIdentityProfile(instanceName,ProfileName,description=None)
```

Argument	Definition
instanceName	Name of the MSAS instance associated with the identity store profile to be modified.
ProfileName	Name of the identity store profile to be modified.

Example

This example selects the identity profile `identity-profile` in MSAS instance `MSAS-123456` for modification:

```
wls:/mydomain/serverConfig>
selectIdentityProfile('MSAS-123456','identity-profile')
```

deleteIdentityProfile

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Delete an identity store profile from a MSAS instance within a repository session.

Issuing this command outside of a repository session will result in an error.

> **Note:** You cannot both create and delete the same identity store profile in a single repository session. You must commit the session after creating a profile, and then start a new session in order to delete that profile.

Syntax

```
deleteIdentityProfile(instanceName,Profilename)
```

Argument	Definition
instanceName	Name of the MSAS instance associated with the identity store profile to be deleted.
ProfileName	Name of the identity store profile to be deleted.

Example

This example deletes the identity profile `identity-profile` from the MSAS instance `MSAS-123456`.

```
wls:/mydomain/serverConfig>
deleteIdentityProfile('MSAS-123456','identity-profile')
```

setIdentityProfileDirectory

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Set or update the directory information for an identity store profile in a MSAS instance within a repository session. The `secure` argument can be used to determine if the identity store connection should be made over SSL.

Before running this command, an identity store profile must be selected for modification or must be created in the current session.

Syntax

```
setIdentityProfileDirectory(DirectoryType,hosts,bindDN,bindPass,baseDN,isSecure)
```

Argument	Definition
directoryType	Type of identity store profile directory. Supported types are: ■ OID (Oracle Internet Directory) ■ OUD (Oracle Unified Directory) ■ ACTIVE_DIRECTORY ■ ODSEE (Oracle Directory Server Enterprise Edition) ■ WLS_LDAP (Embedded LDAP in WebLogic Server)
hosts	Host port information of the directory in the form `host:port`.
bindDN	The distinguished name of the user to be used for connecting to the directory. For example: CN=Administrator,CN=Users,DC=mycompany,DC=com.
bindPass	Bind distinguished name (DN) password for accessing the directory.
baseDN	Base distinguished name (DN) information of the directory. For example: DC=mycompany,DC=com
isSecure	Flag (boolean) to indicate if the connection to the directory should be made over SSL. When set to true the connection is configured over SSL.

Example

This example creates an identity store profile directory with a type of `OID` on a host named ['host.example.com:1234'], with bind distinguished name of `cn=host,dn=oracle,dn=com`, using a password of `welcome`, and a base distinguished name of `cn=host,dn=oracle,dn=com`, which does not require an SSL connection because the `secure` argument is `false`.

```
wls:/mydomain/serverConfig>
setIdentityProfileDirectory('OID',['host.example.com:1234'],'cn=host,dn=oracle,dn=
com','welcome','cn=us,dn=oracle,dn=com',false)
```

This example creates an identity store profile directory that is identical to the one created in the previous example except that it requires an SSL connection to access the directory because the `secure` argument is `true`.

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

```
setIdentityProfileDirectory('OID',['host.example.com:1234'],'cn=host,dn=oracle,dn=
com','welcome','cn=us,dn=oracle,dn=com',true)
```

setIdentityProfileUser

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Set or update the user information for an identity store profile in a MSAS instance within a repository session.

Before running this command, an identity store profile must be selected for modification or must be created in the current session.

Syntax

```
setIdentityProfileUser(baseDN,loginIDAttribute,objectClassNames))
```

Argument	Definition
baseDN	The base distinguished name (DN) used to create users.
loginIDAttribute	The login identity of the user.
objectClassNames	The fully-qualified names of one or more of LDAP object classes used to identify users.

Example

This example creates an identity store profile user with base DN of `cn=host,dn=oracle,dn=com`, with login ID of `uid`, which is represented by a schema object class named `inteorgperson`.

```
wls:/mydomain/serverConfig>
setIdentityProfileUser('cn=user,dn=oracle,dc=com','uid',['inteorgperson'])
```

setIdentityProfileGroup

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Set or update the group information for an identity store profile in a MSAS instance within a repository session.

Before running this command, an identity store profile must be selected for modification or must be created in current session.

Syntax

```
setIdentityProfileGroup(baseDN,groupNameAttribute,objectClassNames)
```

Argument	Definition
baseDN	The base DNs used to create groups or enterprise roles.
groupNameAttribute	The attribute that uniquely identifies the name of the enterprise role or group.
objectClassNames	The fully-qualified names of one or more LDAP object classes used to represent enterprise roles or groups.

Example

This example creates an identity store profile user group with a base DN of `cn=host,dn=oracle,dn=com`, with group name attribute of `cn`, which is represented by a schema object class named `groupofuniquenames`.

```
wls:/mydomain/serverConfig>
setIdentityProfileGroup('cn=group,dn=oracle,dc=com','cn',['groupofuniquenames'])
```

exportMSASAppMetadata

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Exports MSAS application metadata from the repository into a specified ZIP archive. If the specified archive already exists, you can choose whether to merge the documents into the existing archive, overwrite the archive, or cancel the operation. By default, all metadata for MSAS applications in the current domain is exported to the archive. However, you can use a MSAS instance name and MSAS application name to export specific metadata for these MSAS applications in the repository.

Syntax

```
exportMSASAppMetadata(archiveFileName,[instanceName=None],[applicationName=None],[
includeShared='false'])
```

Argument	Definition
archiveFileName	Name of the ZIP archive. If the specified archive already exists, you can choose whether to overwrite the archive, merge the documents into the existing archive, or cancel the operation. During override, the original archive is backed up and a message describes the location of the backup archive.
instanceName	Optional. The name of the MSAS instance from which you want to export the metadata of one or more MSAS applications. If no MSAS instance name is specified, but a valid MSAS application name is specified, then the metadata for all MSAS instances containing this MSAS application is exported. When neither the MSAS instance nor MSAS application name is specified, then the metadata for all the MSAS applications across all MSAS instances is exported. A wildcard '%' is allowed. If this argument is set to `None`, empty '', or '%', then all MSAS instances are searched.
applicationName	Optional. The name of the MSAS application for the metadata to be exported. A wildcard '%' is allowed. If this argument is set to `None`, empty '', or '%', then all applications in the specified instance are searched.
includeShared	Optional. Specifies whether a shared resource (such as a referenced policy) should be included with exported metadata.

Example

This example exports the metadata for all the MSAS applications across all MSAS instances into an archive named `MSASApplications.zip`.

```
wls:/mydomain/serverConfig> exportMSASAppMetadata('/tmp/MSASApplications.zip')
```

This example also exports the metadata for all MSAS applications across all MSAS instances into the `MSASApplications.zip` archive.

```
wls:/mydomain/serverConfig>
exportMSASAppMetadata('/tmp/MSASApplications.zip','',[''])
```

This example exports the metadata for all applications across all MSAS instances that begin with `MSAS` into the `MSASApplications.zip` archive.

```
wls:/mydomain/serverConfig>
exportMSASAppMetadata('/tmp/MSASApplications.zip','MSAS%')
```

This example exports the metadata for all applications that begin with `virtual` on an instance named `MSAS-123456` into the `MSASApplications.zip` archive.

```
wls:/mydomain/serverConfig>
exportMSASAppMetadata('/tmp/MSASApplications.zip','MSAS-1234561',['virtual%'])
```

This example exports the metadata for an application named `Virtual_Foo` on the `MSAS-123456` instance into the `MSASApplications.zip` archive.

```
wls:/mydomain/serverConfig>
exportMSASAppMetadata('/tmp/MSASApplications.zip','MSAS-1234561',['virtual_Foo'])
```

This example exports the metadata for all applications that begin with `virtual` across all MSAS instances into the `MSASApplications.zip` archive.

```
wls:/mydomain/serverConfig>
exportMSASAppMetadata('/tmp/MSASApplications.zip','',['virtual%'])
```

This example exports the metadata for all applications that begin with `virtual`, including application's shared resources, on the `MSAS-123456` instance into the `MSASApplications.zip` archive.

```
wls:/mydomain/serverConfig>
exportMSASAppMetadata('/tmp/MSASApplications.zip','MSAS-1234561',['virtual%'],true
)
```

importMSASAppMetadata

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

Import MSAS application metadata into the repository from a specified ZIP archive. You can use the `map` argument to provide the location of a file that describes how to map physical information from the source environment to the target environment. You can generate a new map file by setting the `generateMapFile` argument to `true`.

Syntax

```
importMSASAppMetadata(archiveFileName,[mapFileName=None],[generateMapFlag='false']
)
```

Argument	Definition
archiveFileName	Name of the ZIP archive to be imported.
mapFileName	Optional. Location of a sample map file that describes how to map physical information from the source environment to the target environment. You can generate a new map file by setting the `generateMapFlag` argument to `true`. If the map file already exists, it will be overwritten; however, no import takes place in the repository. If you specify a map file without setting the `generateMapFlag`, or setting it to `false`, and the map file does not exist, the operation fails and an error is displayed.
generateMapFlag	Optional. Specify whether to generate a map file at the location specified by the `mapFileName` argument. No metadata is imported when this argument is set to `true`, only a map file is generated. The default is `false`.

Example

This example imports application metadata from the `MSASartifacts.zip` archive without using a map file.

```
wls:/mydomain/serverConfig> importMSASAppMetadata('/tmp/MSASartifacts.zip')
```

This example collects all application metadata from the `MSASartifacts.zip` archive and puts it into a map file named `MSASMapfile.txt`. If the specified map file already exists, it will be overwritten.

```
wls:/mydomain/serverConfig>
importMSASAppMetadata('/tmp/MSASartifacts.zip','/tmp/MSASmapfile.txt', true)
```

This example imports application metadata from the specified `MSASartifacts.zip` archive according to the `MSASmapfile.txt` map file.

```
wls:/mydomain/serverConfig>
importMSASAppMetadata('/tmp/MSASartifacts.zip','/tmp/MSASmapfile.txt')
```

migrateMSASAppHostports

Command Category: MSAS Configuration Management

Use with WLST: Online

Description

For applications in the repository that match the specified MSAS instance name and application name, replaces the source host:port values (for example, URLs to a back-end service) with host:port values, according to the source-to-target mapping in the specified `mapFileName`. You can generate a new map file by setting the `generateMapFlag` argument to `true`.

Syntax

```
migrateMSASAppHostports(instanceName,applicationName,mapFileName,[generateMapFlag=
'false'])
```

Argument	Definition
instanceName	Name of the MSAS instance. A wildcard is not accepted.
applicationName	Name of MSAS application whose referenced back-end service host:port information needs to be migrated or replaced. A wildcard '%' is allowed. If this argument is set to `None`, "empty '', or '%', then all applications in the specified MSAS instance are searched.
mapFileName	Location of a input map file that describes how to map source MSAS host:port values to a target host:port. You can generate a new map file by setting the `generateMapFlag` argument to `true`. If the map file already exists, it will be overwritten. The generated map file provides destination host:port values for the source host:port values that need to be replaced. If you specify a map file without setting the `generateMapFlag` argument to `true`, the map file is treated as an input map file and it will be read in for source-to-target host:port replacement for all matched applications. If you specify a map file without setting the `generateMapFlag`, or by setting it to `false`, and the map file does not exist, the operation fails and an error is displayed.
generateMapFlag	Optional. Specify whether to generate a map file at the location specified by the `mapFileName` argument. No MSAS application metadata is modified when this argument is set to `true`, only a map file is generated. The default is `false`.

Example

This example collects all host:port values for the `myApp` application on an instance named `MSAS-1234`, and puts it into a newly-generated map file named `generatedMapfile.txt`. (Note that if a specified map file already exists, it will be overwritten).

```
wls:/mydomain/serverConfig>  migrateMSASAppHostports
('MSAS-1234','myApp','/tmp/generatedMapfile.txt',true)
```

This example migrates the host:port values for all applications that begin with `myApp` on the `MSAS-1234` instance according to the `myMapfile.txt` map file.

```
wls:/mydomain/serverConfig> migrateMSASAppHostports('MSAS-1234', 'myApp%',
'/tmp/myMapfile.txt')
```

The following migrates the host:port values for all applications on the `MSAS-1234` instance according to the `myMapfile.txt` map file.

```
wls:/mydomain/serverConfig> migrateMSASAppHostports('MSAS-1234', 'None',
'/tmp/myMapfile.txt')
```

resetWSMPolicyRepository

Command Category: Policy Repository Management

Use with WLST: Online

> **Caution:** This command will delete all MSAS artifacts, including registered MSAS application and configuration documents. Restarting the server would recover the seed MSAS documents in the repository but will not recover other user-created documents. Therefore, prior to running this command, Oracle recommends running the exportMSASAppMetadata command to back up all your MSAS metadata.

Description

Delete the existing policies stored in the repository and refresh it with the latest set of predefined policies that are provided in the new installation of the Oracle MSAS software. You can use the `clearStore` argument to specify whether to delete all policies, including custom user policies, from the repository before loading the new predefined policies.

> **Note:** In order to reseed the repository with all Oracle MSAS predefined policies, you must restart the Mobile Security Manager (MSM) server after running the `resetWSMPolicyRepository` command. For more information about restarting the MSM server, see "Starting or Stopping the Oracle Stack" in *Oracle Fusion Middleware Installation Guide for Oracle Identity and Access Management*.

Syntax

```
resetWSMPolicyRepository([clearStore='false'])
```

Argument	Definition
clearStore='false'	Policies to be deleted. Valid values are: ■ `true`—All policies in the repository, including custom user policies, are deleted. ■ `false`—Only the predefined policies supplied by Oracle are deleted. The default is `false`.

Example

The following example deletes all the policies in the repository, including user policies, and adds the predefined policies provided in the current product installation:

```
wls:/wls-domain/serverConfig>resetWSMPolicyRepository(true)
```

abortRepositorySession

Command Category: Session

Use with WLST: Online

Description

Abort the current Oracle Repository modification session, discarding any changes that were made to the repository during the session.

Syntax

```
abortRepositorySession()
```

Examples

The following example aborts the current session.

```
wls:/wls-domain/serverConfig>abortRepositorySession()
```

beginRepositorySession

Command Category: Session

Use with WLST: Online

Description

Begin a session to modify the Oracle Repository. A repository session can only act on a single document. An error will be displayed if there is already a current session.

Syntax

```
beginRepositorySession()
```

Example

The following example begins a session.

```
wls:/wls-domain/serverConfig>beginRepositorySession()
```

commitRepositorySession

Command Category: Session

Use with WLST: Online

Description

Write the contents of the current session to the Oracle Repository. Messages are displayed that describe what was committed. An error will be displayed if there is no current session.

Syntax

```
commitRepositorySession()
```

Example

The following example commits the current repository modification session.

```
wls:/wls-domain/serverConfig>commitRepositorySession()
```

describeRepositorySession

Command Category: Session

Use with WLST: Online

Description

Describe the contents of the current session. This will either indicate that the session is empty or list the name of the document that is being updated, along with the type of update (create, modify, or delete). An error will be displayed if there is no current session.

Syntax

```
describeRepositorySession()
```

Examples

The following example describes the current session.

```
wls:/wls-domain/serverConfig>describeRepositorySession()
```

createWSMTokenIssuerTrustDocument

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Within a session, create a new token issuer trust document using the name provided. A display name can also be provided as the second argument.

You must start a session (`beginWSMSession`) before creating or modifying any token issuer trust documents. If there is no current session or there is already an existing modification process, an error is displayed.

For more information on using this command, see "Configuring Trusted Issuers and DN Lists Using WLST" in *Administering Mobile Security Access Server*.

Syntax

```
createWSMTokenIssuerTrustDocument(name, displayName)
```

Arguments	Definition
name	Name of the document to be created. An error is thrown if a name is not provided If a document by this name already exists, then a new document will not be created.
displayName	Optional. Display name for the document.

Examples

In the following example, the trust document named `tokenissuertrustWLSbase_domain` is created, with a display name of `wls_domain Trust Document`. In the second example, no display name is provided.

```
wls:/wls-domain/serverConfig>
createWSMTokenIssuerTrustDocument("tokenissuertrustWLSbase_domain","wls_domain
Trust Document")
wls:/wls-domain/serverConfig>
createWSMTokenIssuerTrustDocument("tokenissuertrustWLSbase_domain")
```

deleteWSMTokenIssuerTrust

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Within a session, delete a trusted token issuer and its associated trusted DN list. Supported values for a SAML assertion or JWT token type are `dns.sv`, `dns.hok`, or `dns.jwt`. This issuer must exist in the token issuer trust document selected in the session for modification. If no trusted key identifiers exist, then the issuer itself is deleted.

To delete a specified list of trusted key identifiers for an issuer, use selectWSMTokenIssuerTrustDocument.

You must start a session (`beginWSMSession`) and select a token issuer trust document for modification before executing this command. If there is no current session or there is already an existing modification process, an error is displayed.

You cannot modify the default token issuer trust document.

Syntax

```
deleteWSMTokenIssuerTrust(type, issuer)
```

Arguments	Definition
type	The type of SAML assertion or JWT tokens the trusted issuer issues: ■ `dns.sv` – SAML sender vouches client list ■ `dns.hok` – SAML HOK or Bearer ■ `dns.jwt` – JWT token.
issuer	The name of the issuer whose trusted DN list will be deleted (for example, SAML assertion or JWT token). The issuer will also be deleted.

Examples

In the following example, the issuer `www.yourCompany.com` and the DN list in the `dns.sv` trusted SAML sender vouches client list for the issuer are deleted:

```
wls:/wls-domain/serverConfig> deleteWSMTokenIssuerTrust('dns.sv',
'www.yourCompany.com')
```

In the following example, the issuer `www.yourCompany.com` and the DN list in the `dns.jwt` trusted JWT token sender vouches client list for the issuer are deleted:

```
wls:/wls-domain/serverConfig> deleteWSMTokenIssuerTrust('dns.jwt ',
'www.yourCompany.com')
```

deleteWSMTokenIssuerTrustAttributeRule

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Delete a token attribute rule associated with a trusted DN from the token issuer trust document.

To delete only the list of filter values for an attribute, use the setWSMTokenIssuerTrustAttributeFilter command.

You must start a session (`beginWSMSession`) and select a token issuer trust document for modification before executing this command. If there is no current session or there is already an existing modification process, an error is displayed.

Syntax

```
deleteWSMTokenIssuerTrustAttributeRule(dn)
```

Arguments	Description
dn	The DN of the token signing certificate that identifies the rule to be deleted.

Examples

In the following example, the token attribute rule associated with the `'CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US` trusted DN is deleted.

```
wls:/wls-domain/serverConfig> deleteWSMTokenIssuerTrustAttributeRule('CN=weblogic,
OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US')
```

deleteWSMTokenIssuerTrustDocument

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Deletes the specified token issuer trust document permanently from the repository. The default token issuer trust document (`oracle-default`) cannot be deleted.

Syntax

```
deleteWSMTokenIssuerTrustDocument (name)
```

Arguments	Definition
name	Name of the token issuer trust document to be deleted.

Examples

In the following example, the token issuer trust document `tokenissuertrustWLSbase_domain` trust document is deleted:

```
wls:/wls-domain/serverConfig>
deleteWSMTokenIssuerTrustDocument('tokenissuertrustWLSbase_domain')
```

displayWSMTokenIssuerTrust

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Display a trusted token issuer and its associated trusted DN list. Supported values for a SAML assertion or JWT token type are `dns.hok`, `dns.sv`, or `dns.jwt`. The `issuer` argument is optional. If the issuer and type is specified and exists in the trusted issuer list for the type, then the associated DN lists for the issuer is displayed. If `issuer` is not set, then all trusted issuers of the given type are listed.

Syntax

```
displayWSMTokenIssuerTrust(type, issuer)
```

Arguments	Definition
type	The type of SAML assertion or JWT tokens the trusted issuer issues: ■ `dns.sv` – SAML sender vouches client list ■ `dns.hok` – SAML HOK or Bearer ■ `dns.jwt` – JWT token.
issuer	Optional. The issuer whose trusted DN list is displayed (for example, SAML assertion or JWT token). If not set, the list of all the trusted issuers is displayed.

Examples

In the following example, the DN lists in the `dns.sv` trusted SAML sender vouches client list for the `www.oracle.com` trusted issuer are displayed:

```
wls:/wls-domain/serverConfig>displayWSMTokenIssuerTrust('dns.sv',
'www.oracle.com')
```

In the following example, the names of all trusted SAML issuers associated with the `dns.sv` trusted SAML sender vouches client list are displayed:

```
wls:/wls-domain/serverConfig>displayWSMTokenIssuerTrust('dns.sv', None)
```

exportWSMTokenIssuerTrustMetadata

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Export all the trust configurations (issuer, DNs, and token attribute rules) for all trusted issuers. The trust configuration will be exported to an XML file identified by the specified location. The trust configuration for the issuers specified in the exclude list will not be exported. If no argument is passed, the trust configuration for all trusted issuers will be exported.

Syntax

```
exportWSMTokenIssuerTrustMetadata(trustFile,excludeIssuers=None)
```

Arguments	Definition
trustFile	The location of the file where the exported metadata will be stored.
excludeIssuers	Optional. The list of issuers for which trust configuration should not be exported.

Examples

In the following example, all trusted issuer configurations are exported to the specified XML file except for `www.oracle.com` and `www.yourcompany.com`, which have been excluded:

```
wls:/wls-domain/serverConfig>exportWSMTokenIssuerTrustMetadata(trustFile='/tmp/tru
stData.xml',['www.oracle.com','www.myissuer.com'])

Starting Operation exportWSMTokenIssuerTrustMetadata ...
Configuration for trusted issuers successfully exported.
```

In the following example, all specified trusted issuer configurations are exported to the specified XML file:

```
wls:/wls-domain/serverConfig>exportWSMTokenIssuerTrustMetadata(trustFile='/tmp/tru
stData.xml')

Starting Operation exportWSMTokenIssuerTrustMetadata ...
Configuration for trusted issuers successfully exported.
```

importWSMTokenIssuerTrustMetadata

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Import the trust configurations (issuers, DNs, and token attribute rules) for all trusted issuers. The trust configuration will be imported from an XML file identified by the specified location.

Syntax

```
importWSMTokenIssuerTrustMetadata(trustFile)
```

Arguments	Definition
trustFile	The location of the file where the imported metadata will be stored.

Examples

In the following example, all trusted issuer configurations are imported from the specified XML file:

```
wls:/wls-domain/serverConfig>importWSMTokenIssuerTrustMetadata(trustFile='/tmp/tru
stData.xml')

Starting Operation importWSMTokenIssuerTrustMetadata ...
Configuration for trusted issuers successfully imported.
```

listWSMTokenIssuerTrustDocuments

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

When used without any arguments, this command lists all the token issuer trust documents in the repository. If the detail argument is set to `true`, the display name and the status of the document are also displayed.

You can use the wildcard character (*) in combination with other characters. If no wildcard character is specified in the name argument, the document that matches the name argument exactly is displayed. If the `detail` argument is set to `true`, the contents of the document are listed.

This command can be executed inside and outside of a session.

Syntax

```
listWSMTokenIssuerTrustDocuments(name=None, detail='false')
```

Arguments	Definition
name	Optional. Name of the token issuer trust document. You can use wildcards with this argument.
detail	Optional. List the details for the requested document. The default is `false`.

Examples

In the following example, the token issuer trust document `tokenissuertrustWLSbase_domain` trust document is listed along with any details:

```
wls:/wls-domain/serverConfig>
listWSMTokenIssuerTrustDocuments(tokenissuertrustWLSbase_domain,'true')
```

revokeWSMTokenIssuerTrust

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Revokes trust by removing all trusted issuers and associated configurations (DNs and token attribute rules). The issuers specified in the optional exclude list will not be removed. If no argument is passed, then all trusted issuers and the associated configuration are removed.

Syntax

```
revokeWSMTokenIssuerTrust(excludeIssuers=None)
```

Arguments	Definition
excludeIssuers	Optional. The list of issuers for which trust configuration should not be removed.

Examples

In the following example, all trusted issuer configurations are removed except for `www.oracle.com` and `www.yourcompany.com`, which have been excluded:

```
wls:/wls-domain/serverConfig>revokeWSMTokenIssuerTrust(['www.oracle.com','www.your
company.com'])

Starting Operation revokeWSMTokenIssuerTrust ...
Configuration for trusted issuers successfully removed.
```

In the following example, all trusted issuer configurations are removed:

```
wls:/wls-domain/serverConfig>revokeWSMTokenIssuerTrust()

Starting Operation revokeWSMTokenIssuerTrust ...
Configuration for trusted issuers successfully removed.
```

selectWSMTokenIssuerTrustDocument

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Selects the token issuer trust document, identified by the name argument, to be modified in the session. The name must match the value of the name attribute in the document.

You must start a session (`beginWSMSession`) before executing this command. If there is no current session or there is already an existing modification process, an error is displayed.

You cannot modify the default token issuer trust document.

Syntax

```
selectWSMTokenIssuerTrustDocument(name)
```

Argument	Definition
name	Name of the document to modified in the session. An error is thrown if a name is not provided.

Examples

In the following example, the `tokenissuertrustWLSbase_domain` document is selected for modification:

```
wls:/wls-domain/serverConfig>
selectWSMTokenIssuerTrustDocument('tokenissuertrustWLSbase_domain')
```

setWSMTokenIssuerTrust

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Configure a trusted token issuer and define trusted keys or a trusted DN list for the issuer. Supported values for a SAML assertion or JWT token type are `dns.hok`, `dns.sv`, or `dns.jwt`. The `trustedKeyIDs` argument is optional. If you do not set this argument, only the trusted issuer will be set for the specified type.

This command can be used to specify the DN list associated with a trusted token issuer, update the list, or delete the list. See the following examples.

Syntax

```
setWSMTokenIssuerTrust(type, issuer, trustedKeyIDs)
```

Argument	Definition
issuer	The name of the trusted issuer, for example `www.oracle.com`.
type	The type of SAML assertion or JWT tokens the trusted issuer issues: ■ `dns.sv` – SAML sender vouches client list ■ `dns.hok` – SAML HOK or Bearer ■ `dns.jwt` – JWT token.
trustedKeyIDs	Optional. An array of DNs for token signing certificates associated with the issuer for the specified type. This is a comma-separated list with the format `['CN=name1', 'CN=name2', 'CN=name3', ...]`. If you enter an empty set (`[]`) the list of DN values will be deleted for the issuer.

Examples

In the following example, `CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US'` is set as a DN in the `dns.sv` DN list for the `www.oracle.com` trusted issuer:

```
wls:/wls-domain/serverConfig>setWSMTokenIssuerTrust('dns.sv', 'www.oracle.com',
['CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US'])
```

In the following example, the name `CN=orcladmin, OU=Doc, O=Oracle, C=US'` in added to the `dns.sv` DN list for the `www.oracle.com` trusted issuer:

```
wls:/wls-domain/serverConfig>setWSMTokenIssuerTrust('dns.sv','www.oracle.com',
['CN=weblogic, OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US',
'CN=orcladmin, OU=Doc, O=Oracle, C=US'])
```

In the following example, the list of DN values in the `dns.sv` DN list is removed from the `www.oracle.com` trusted issuer:

```
wls:/wls-domain/serverConfig>setWSMTokenIssuerTrust('dns.sv', 'www.oracle.com',
[])
```

setWSMTokenIssuerTrustAttributeFilter

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Specify token attribute filtering rules for a trusted DN list. For each trusted DN configured for an issuer, a token attribute filtering rule can be configured and applied. Each rule has two parts: a name ID and an attributes part for attributes in a SAML assertion or a JWT token. The name ID and each attribute can contain a filter with multiple value patterns.

To remove the list of filters for an attribute for the signing certificate, use an empty set ([]) for the value of `filters`.

Note: You must first use the `setWSMTokenIssuerTrust` command to configure a list of trusted DN names for an issuer.

Syntax

```
setWSMTokenIssuerTrustAttributeFilter(dn, attrName, filters)
```

Argument	Definition
dn	The DN of the token signing certificate. The DN as the identifier of the token attribute rule where modifications would be done.
attrName	The name of the user attribute for which the filtering will be applied. The value can be as follows: ■ `name-id`—assert a subject name ID.
filters	Optional. List of filters for the attribute. The list has the format ['`value1`', '`value2`', '`value3`, ... Each value can be an exact name or a name pattern with a wildcard character "`*`". When `name-id` is selected for the `attrName` argument, then the value of the subject name ID in the incoming SAML assertion must match one of the specified values to go through. If no values are specified, then any value for the subject name ID will go through.

Examples

In the following example, the name ID `yourTrustedUser` is set as a trusted user for the `weblogic` trusted DN:

```
wls:/wls-domain/serverConfig> setWSMTokenIssuerTrustAttributeFilter('CN=weblogic,
OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US','name-id',
['yourTrustedUser'])
```

In the following example, the name IDs `jdoe` is added to the list of trusted users for the `weblogic` trusted DN:

```
wls:/wls-domain/serverConfig> setWSMTokenIssuerTrustAttributeFilter('CN=weblogic,
OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US','name-id',
['yourTrustedUser', 'jdoe'])
```

In the following example, the list of trusted users for the `weblogic` trusted DN is removed:

```
wls:/wls-domain/serverConfig> setWSMTokenIssuerTrustAttributeFilter('CN=weblogic,
OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US', 'name-id', [])
```

setWSMTokenIssuerTrustAttributeMapping

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Specify token attribute mapping rules for a trusted DN list. For each trusted DN configured for a token issuer, a token attribute mapping rule can be configured and applied. Each rule has two parts: a name ID and an attributes part for attributes associated with a SAML assertion or a JWT token.

For a trusted DN, a token attribute mapping rule sets the mapping for the value of an attribute as specified by the `attrName` argument. The `userAttribute` argument is optional and indicates the local user attribute it corresponds to. The `userMappingAttribute` argument is optional and indicates the user attribute to be used in the system to authenticate the users. If the attribute as identified by `attrName` exists for a token attribute rule for the DN, the mapping is overwritten by the new value.

For example, in federated environments, where the user subject ID (for example, `mail`) in the token is different from the user attribute (for example, `uid`) for authenticating the same user, the name ID and each attribute can map the local user attribute for the subject name ID to the local user attribute to authenticate a trusted user.

Note: You must first use the `setWSMTokenIssuerTrust` command to configure a list of trusted DN names for an issuer.

Syntax

```
setTokenIssuerTrustAttributeMapping(dn, attrName, userAttribute=None,
userMappingAttribute=None)
```

Arguments	Description
dn	The trusted DN of a token signing certificate.
attrName	The name of the use attribute for which the mapping will be applied The value can be as follows: ■ `name-id`
userAttribute	Optional. The local name of the user attribute in the local identity store that the subject name ID corresponds to. The value can be as follows: ■ `mail`
userMappingAttribute	Optional. The value of the local name of the user attribute in the local identity store that the subject name ID maps to for authentication. The value can be as follows: ■ `uid`

Examples

In the following example, the `mail` attribute for the Subject ID in the token is mapped to the `uid` attribute.

```
wls:/base_domain/serverConfig>setTokenIssuerTrustAttributeMapping('CN=weblogic,
OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US', 'name-id', 'mail',
```

```
'uid')

Starting Operation setWSMTokenIssuerTrustAttributeMapping ...
The token attribute mapping are successfully set
```

In the following example, the local user attribute for the Subject ID in the token is mapped to the `uid` attribute.

```
wls:/base_domain/serverConfig>setTokenIssuerTrustAttributeMapping('CN=weblogic,
OU=Orakey Test Encryption Purposes Only, O=Oracle, C=US', 'name-id', '', 'uid')
```

setWSMTokenIssuerTrustDisplayName

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

Description

Sets or resets the display name of the Token Issuer Trust document currently selected in the session.

You must start a session (`beginWSMSession`) before creating or modifying any token issuer trust documents. If there is no current session or there is already an existing modification process, an error is displayed.

Syntax

```
setWSMTokenIssuerTrustDisplayName("displayName")
```

Arguments	Definition
displayName	Name to be set as a display name for the document currently selected for modification in the session.

Examples

In the following example, the display name for the trust document being modified is set to `Test Document`.

```
wls:/wls-domain/serverConfig> setWSMTokenIssuerTrustDisplayName("Test Document")
```

checkWSMStatus

Command Category: Diagnostic

Use with WLST: Online

Description

Check the status of the Oracle components that are required for proper functioning of the product. The Oracle components that are checked are the policy manager (`wsm-pm`), the agent (`agent`), and the credential store and keystore configuration. The status of the components can be checked together or individually.

> **Note:** The Policy Manager (`wsm-pm`) application must be deployed and running for the check status tool to function correctly.

Syntax

```
checkWSMStatus([component=None],[address=None],[verbose=true])
```

Arguments	Description
component	Optional. All checks will be performed if no value is specified. Valid options are: ■ `wsm-pm`—Policy Manager. Checks the configuration state of the policy manager component. ■ `agent`—Enforcement Agent. Checks status of end-to-end service-side enforcement through the `wsm` agent component. The enforcement check is specific only to the environment from which the command is run. ■ `credstore`—Credential Store. Checks whether the credentials are configured for the keystore password, signing, and encryption certificates in the keystore.
address	Optional. The HTTP URL of the host running the Policy Manager `wsm-pm` application. This value is required for checking enforcement through an agent component, for example, `checkWSMStatus('agent', 'http://localhost:7001')` The address is not required in the WebLogic Server domain where auto-discovery is present.
verbose	Optional. If the value of this flag is `true`, then the detailed messages (including stack trace, if any) are displayed. Default is `false`.

Examples

In the following example, the `checkWSMStatus` command is run without arguments. The status of the credential store, policy manager, and enforcement agent is returned.

```
wls:/base_domain/serverConfig> checkWSMStatus()

Credential Store Configuration:

PASSED.
        Message(s):
             keystore.pass.csf.key : Property is configured and its value is
"keystore-csf-key".
```

```
                Description: The "keystore.pass.csf.key" property points to the
CSF alias that is mapped to the username and password of the keystore. Only the
password is used; username is redundant in the case of the keystore.
            keystore-csf-key : Credentials configured.
            keystore.sig.csf.key : Property is configured and its value is
"sign-csf-key".
                Description: The "keystore.sig.csf.key" property points to the
CSF alias that is mapped to the username and password of the private key that is
used for signing.
            sign-csf-key : Credentials configured.
            Sign Key : Key configured.
                Alias - orakey
            Sign Certificate : Certificate configured.
                Alias - CN=weblogic, OU=Orakey Test Encryption Purposes Only,
O=Oracle, C=US
                Expiry - June 28, 2020 11:17:12 AM PDT
            keystore.enc.csf.key : Property is configured and its value is
"enc-csf-key".
                Description: The "keystore.enc.csf.key" property points to the
CSF alias that is mapped to the username and password of the private key that is
used for decryption.
            enc-csf-key : Credentials configured.
            Encrypt Key : Key configured.
                Alias - orakey
            Encrypt Certificate : Certificate configured.
                Alias - CN=weblogic, OU=Orakey Test Encryption Purposes Only,
O=Oracle, C=US
                Expiry - June 28, 2020 11:17:12 AM PDT

Policy Manager:


PASSED.
        Message(s):
             OWSM Policy Manager connection state is OK.
             OWSM Policy Manager connection URL is "host.example.com:1234".

Enforcement Agent:


PASSED.
        Message(s):
             Enforcement is successful.
             Service URL:
http://host.example.com:7001/Diagnostic/DiagnosticService?wsdl
```

In the following example, the credential store key `keystore-csf-key` is deleted and the `checkWSMStatus` command is rerun for the credential store `credstore`. The status check fails because the csf-key `keystore-csf-key` is not present in the credential store:

```
wls:/base_domain/serverConfig> deleteCred(map="oracle.wsm.security",
key="keystore-csf-key")
wls:/base_domain/serverConfig> checkWSMStatus('credstore')

Credential Store Configuration:

FAILED.
        Message(s):
             keystore.pass.csf.key : Property is configured and its value is
"keystore-csf-key".
```

```
                Description: The "keystore.pass.csf.key" property points to the
CSF alias that is mapped to the username and password of the keystore. Only the
password is used; username is redundant in the case of the keystore.
             keystore-csf-key : Credentials not configured.

Credential Store Diagnostic Messages:
        Message(s):
                The csf-key keystore-csf-key is not present in the credential
store.

 Perform the following steps to update the credential store (using WLST
commands):-
 1. connect()
 2. createCred(map="oracle.wsm.security", key="keystore-csf-key",
user="keystore-csf-key", password="<keystore-password>", desc="Keystore Password
CSF Key")
 NOTE:- All the above commands are based on the Domain level configurations. The
actual csf key may be overridden at runtime due to config override. See
Documentation for more details.
```

In the following example, the csf-key `keystore-csf-key` is configured and the `checkWSMStatus` command is rerun. The configuration check passes.

```
wls:/base_domain/serverConfig> createCred(map="oracle.wsm.security",
key="keystore-csf-key", user="keystore-csf-key", password="welcome1",
desc="Keystore Password CSF Key")
Already in Domain Runtime Tree

wls:/base_domain/serverConfig> checkWSMStatus('credstore')

Credential Store Configuration:


PASSED.
        Message(s):
             keystore.pass.csf.key : Property is configured and its value is
"keystore-csf-key".
                Description: The "keystore.pass.csf.key" property points to the
CSF alias that is mapped to the username and password of the keystore. Only the
password is used; username is redundant in the case of the keystore.
             keystore-csf-key : Credentials configured.
             keystore.sig.csf.key : Property is configured and its value is
"sign-csf-key".
                Description: The "keystore.sig.csf.key" property points to the
CSF alias that is mapped to the username and password of the private key that is
used for signing.
             sign-csf-key : Credentials configured.
             Sign Key : Key configured.
                 Alias - orakey
             Sign Certificate : Certificate configured.
                 Alias - CN=weblogic, OU=Orakey Test Encryption Purposes Only,
O=Oracle, C=US
                 Expiry - June 28, 2020 11:17:12 AM PDT
             keystore.enc.csf.key : Property is configured and its value is
"enc-csf-key".
                Description: The "keystore.enc.csf.key" property points to the
CSF alias that is mapped to the username and password of the private key that is
used for decryption.
             enc-csf-key : Credentials configured.
             Encrypt Key : Key configured.
                 Alias - orakey
```

```
            Encrypt Certificate : Certificate configured.
                Alias - CN=weblogic, OU=Orakey Test Encryption Purposes Only,
O=Oracle, C=US
                Expiry - June 28, 2020 11:17:12 AM PDT
true
```

The following example checks the enforcement status of the agent component at the URL `http://localhost:7001`.

```
wls:/test_domain1/serverConfig> checkWSMStatus('agent','http://localhost:7001')

Enforcement Agent:

Note: Enforcement might succeed if OWSM Policy Manager is down due to policy
caching. For such scenarios wsm-pm test must be run prior to this test.

PASSED.
        Message(s):
            Enforcement is successful.
            Service URL: http://localhost:7001/Diagnostic/DiagnosticService?wsdl
```