**Oracle® Fusion Middleware**

WLST Command Reference for Infrastructure Components

12*c* (12.2.1)

**E55587-01**

October 2015

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

ORACLE®

Oracle Fusion Middleware WLST Command Reference for Infrastructure Components, 12*c* (12.2.1)

E55587-01

# Contents

# Preface

This guide describes the Fusion Middleware Infrastructure commands that are available to use with the WebLogic Scripting Tool (WLST).

## Audience

This document is intended for administrators and developers who are configuring Oracle Fusion Middleware or developing applications and want to use the WLST commands for Oracle Fusion Middleware Infrastructure components.

## 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 documentation set:

- *WLST Command Reference for WebLogic Servers*

- *Understanding the WebLogic Scripting Tool*

- *Infrastructure Security WLST Command Reference*

## 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. |

| Convention | Meaning |
| --- | --- |
| *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. |

# What's New in This Guide

The following topics introduce the new and changed features of the custom WLST commands for Oracle Fusion Middleware Infrastructure components.

## New and Changed Features for 12*c* (12.2.1)

The following topics introduce new and changed features for this release:

- The following summarize the changes to the Web services WLST commands.

  - Redesigned Web services WLST framework to provide consistency across all Web service stacks. As a result, there are a number of new and deprecated WLST commands for Oracle Infrastructure Web services. Deprecated commands are identified throughout this document. For a complete list of deprecated commands and the12*c* equivalents, see "Deprecated Commands for Oracle Infrastructure Web Services" in *Release Notes for Oracle Fusion Middleware Infrastructure*.

  - New Web service WLST command categories, including:

    * Session commands for managing a session.

    * Policy Subject commands for viewing and managing Web service and client policy subjects.

    * Configuration commands for viewing and managing OWSM domain configuration.

    For more information, see "Web Services WLST Command Categories" on page 3-10.

  - Command syntax used to identify a policy subject has changed. For more information, see "Specifying Application, Composite, and Service Names" on page 3-4 and "Identifying the Policy Subject" on page 3-6.

  - New OWSM repository WLST commands are provided for exporting application metadata, migrating policy attachments and roles, and upgrading the OWSM repository. For more information, see "OWSM Repository Management Commands" on page 3-140.

  - New WLST commands for managing Web service token issuer trust documents are provided, as described in "Token Issuer Trust Configuration Commands" on page 3-158.

- WLST commands for Oracle HTTP Server are now located in "Oracle HTTP Server WLST Custom Commands" in the *Administrator's Guide for Oracle HTTP Server*.

- New custom WLST commands for DMS parameter-scoped metric rules. For more information, see "DMS Parameter-Scoped Metrics Rules Commands" on page 6-14.

- Globally attached policies are now supported with Java EE web services. As a result, a number of notes have been removed that said that globally attached policies were not supported with Java EE web services. In addition, a number of sections under "Policy Subject Commands" on page 3-16 have been updated to reflect this new support.

- New custom WLST commands for creating, listing, and deleting Web Services connections for ADF Applications. For more information, see "createWebServiceConnection" on page 5-14.

# 1

# Introduction and Roadmap

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

- Document Scope and Audience
- Related Documentation
- Invoking the WebLogic Scripting Tool (WLST)

## 1.1 Document Scope and Audience

This document describes the custom WLST commands for Oracle Fusion Middleware Infrastructure components and services, which include:

- Dynamic Monitoring Service (DMS)
- Logging
- Oracle Application Development Framework (ADF)
- Oracle Fusion Middleware Diagnostic Framework
- Oracle JRF
- Oracle infrastructure web services
- Oracle Metadata Services (MDS)
- Oracle User Messaging Service

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

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

## 1.2 Related Documentation

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

For information about the other WLST commands and other WebLogic Server management interfaces, see:

- *WLST Command Reference for WebLogic Server* describes the WLST commands for WebLogic Server.

- *WLST Command Reference for SOA Suite* describes the WLST commands that are available for Oracle SOA Suite and Oracle Business Process Management (BPM).

- *WebCenter WLST Command Reference* describes the WLST commands that are available for WebCenter components, including WebCenter Portal, WebCenter Content, WebCenter Information Rights Management (IRM), and WebCenter Imaging Process Management (IPM).

- "Oracle HTTP Server WLST Custom Commands" in the *Administrator's Guide for Oracle HTTP Server.*

- *WebLogic Scripting Tool Command Reference for Oracle Traffic Director* describes the WLST commands that are available for Oracle Traffic Director.

- "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 Using JMX for Oracle WebLogic Server* describes using Java Management Extensions (JMX) APIs to monitor and modify WebLogic Server resources.

- *Monitoring Oracle WebLogic Server with SNMP* describes using Simple Network Management Protocol (SNMP) to monitor WebLogic domains.

## 1.3  Invoking the WebLogic Scripting Tool (WLST)

You invoke WLST from the following location:

```
(UNIX) ORACLE_HOME/oracle_common/common/bin/wlst.sh
(Windows) ORACLE_HOME\oracle_common\common\bin\wlst.cmd
```

# 2

# Oracle JRF Custom WLST Commands

Oracle JRF (Java Required Files) consists of those components not included in the WebLogic Server installation that provide common functionality for Oracle business applications and application frameworks. This chapter provides detailed descriptions of custom WLST commands for Oracle JRF, including command syntax, arguments and command examples.

Oracle JRF consists of a number of independently developed libraries and applications that are deployed into a common location. The following components are considered part of Oracle JRF: Oracle Application Development Framework, Oracle Fusion Middleware Audit Framework, Dynamic Monitoring Service, Fabric Common, HTTP Client, Infrastructure Security, Java Object Cache, JMX Framework, JPS, logging, MDS, OJSP.Next, Oracle Web Services, Oracle Web Services Manager, Oracle TopLink, UCP, XDK.

## 2.1 Oracle JRF Commands

Use the commands in Table 2–1 to configure a Managed Server or cluster with Oracle JRF applications and services or to copy the applications and services from one Managed Server or cluster and apply them to another Managed Server or cluster.

In the Use with WLST column, online means the command can only be used when connected to a running server. Offline means the command can only be used when not connected to a running server. Online or offline means the command can be used in both situations.

*Table 2–1  JRF Commands*

| Use this command... | To... | Use with WLST... |
|---------------------|-------|------------------|
| applyJRF | Configures a Managed Server or cluster with Oracle JRF applications and services. | Online or Offline |
| cloneDeployments | Copies the applications and services from Managed Server or cluster and applies them to another Managed Server or cluster. | Online or Offline |

## applyJRF

Use with WLST: Online or Offline

### Description

Configures a Managed Server or cluster with Oracle JRF. Managed Servers that are added by product templates during the template extension process do not need to be explicitly configured with JRF using this command.

Use the applyJRF command when additional Managed Servers or clusters are added to a domain after it is initially extended with a product template. The applyJRF command is required any time you add a Managed Server to a JRF-only domain, or if you add a Managed Server that has been configured for JRF to a domain that contains other Oracle products.

### Syntax

```
applyJRF(target, [domainDir], [shouldUpdateDomain])
```

| Argument | Definition |
| --- | --- |
| target | The name of the Managed Server or cluster to be configured with JRF applications and services. |
| | A value of an asterisk (*) for the target indicates that all clusters and standalone Managed Servers should be configured with JRF. |
| domainDir | The absolute path of the WebLogic Server domain. |
| shouldUpdateDomain | An optional boolean flag that controls how domain updates are carried out. When you set it to true (the default), the function implicitly invokes the following offline commands: readDomain() and updateDomain(), or the online commands: edit(), startEdit(), save(), and activate(). |
| | When you set it to false, you must call WLST commands to update the domain. |

### Example

The following example configures the Managed Server server1 with JRF:

```
wls:/offline> applyJRF('server1', '/my_path/user_templates/domains/my_domain')
```

## cloneDeployments

Use with WLST: Online or Offline

### Description

Replicates all deployments targeted to a particular Managed Server or cluster on a second Managed Server or cluster. This command is provided as a convenience to configure a new Managed Server or cluster so that it has the same deployments as a pre-existing Managed Server or cluster.

The cloneDeployments command does not create new Managed Servers, and it does not copy properties other than deployment information to the target Managed Server.

### Syntax

```
cloneDeployments(domain, source, target, [shouldUpdateDomain])
```

| Argument | Definition |
|---|---|
| domain | The absolute path of the WebLogic Server domain. Ignored if the domain has been read, or if connected in online mode. |
| source | The name of the Managed Server or cluster from which you want to clone deployments. This must be the name of a valid Managed Server or cluster. |
| target | The target Managed Server or cluster that will receive the source server's applications and services. The target Managed Server must already exist. |
| shouldUpdateDomain | An optional boolean flag that controls how domain updates are carried out. When you set it to true (the default), the function implicitly invokes the following offline commands: readDomain() and updateDomain(), or online commands: edit(), startEdit(), save(), and activate(). When you set it to false, you must call WLST commands to update the domain. |

### Example

The following example replicates the deployments from sourceServer to destinationServer:

```
wls:/offline> cloneDeployments( '/my_path/user_templates/domains/my_domain',
 'sourceServer','destinationServer', 'false')
```

# 3

# Web Services Custom WLST Commands

This chapter describes the WebLogic Scripting Tool (WLST) commands for Oracle Infrastructure web services (which includes SOA composites, ADF Business Components, Java EE web services, and RESTful web services. You can use these commands to manage web services from the command line.

> **Note:** Only a subset of the custom WLST commands described in this chapter are supported for Java EE web services.
>
> A subset of WLST commands have been deprecated for Oracle Infrastructure web services and clients. For a complete list of deprecated commands, see "Deprecated Commands for Oracle Infrastructure Web Services" in *Release Notes for Oracle Fusion Middleware Infrastructure*.

For additional details about using these WLST commands for web services, see the following documents:

- *Administering Web Services*.
- *Securing Web Services and Managing Policies with Oracle Web Services Manager*

> **Notes:** To use the Web Services custom WLST commands, you must invoke WLST from the Oracle Common home directory. See "Using Custom WLST Commands" in the *Administering Oracle Fusion Middleware.*
>
> To display the help for the web service and client management and Java EE web service policy management commands, connect to a running instance of the server and enter `help('WebServices')`.
>
> To display the help for the remaining commands, connect to a running instance of the server and enter `help('wsmManage')`.

This chapter contains the following topics:

- Overview of Web Services WLST Commands
- Session Commands
- Policy Subject Commands
- Configuration Commands
- Diagnostic Commands

- Web Service and Client Management Commands
- Policy Management Commands
- Policy Set Management Commands
- OWSM Repository Management Commands
- Token Issuer Trust Configuration Commands
- Secure Conversation Session Management Commands
- JKS Keystore Configuration Commands

## Overview of Web Services WLST Commands

You can use the web services WLST commands, in online mode, to:

- Perform web service configuration and OWSM policy management tasks.

- Manage the OWSM repository.

- Check the status of OWSM components.

- View and define trusted issuers and DN lists for SAML signing certificates.

> **Note:** Ensure that the user is mapped to the appropriate OWSM logical roles, based on the WLST operations you wish to perform. For more information, see "Modifying the User's Group or Role" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

The web services WLST configuration and policy management commands perform many of the same management functions that you can complete using Fusion Middleware Control, such as managing deployed, active, and running web services applications. They can be executed everywhere in WLST online mode, for example:

```
wls:/domain/serverConfig
wls:/domain/domainRuntime
```

The following sections provide more information about using the WLST commands:

- "Specifying Application, Composite, and Service Names"

- "Identifying the Policy Subject"

- "Web Services WLST Command Categories"

## Specifying Application, Composite, and Service Names

The web service WLST commands configure a web service for a specific application. Therefore, the application path name has to uniquely identify the application and the server instance to which it is deployed.

The following sections describe how to specify the application and service names to uniquely identify the web service.

- "Specifying a Web Service Application Name" on page 3-4
- "Specifying a Service Name" on page 3-4

## Specifying a Web Service Application Name

To specify a web service application in a WLST command, use the following format:

```
[/domain/server/]application[#version_number]
```

Parameters shown in brackets [] are optional. The following examples show the sample format for a web service application name:

```
/base_domain/AdminServer/HelloWorld#1_0
/base_domain/server1/HelloWorld#1_0
```

If there is only one deployed instance of an application in a domain, you may omit the domain/server parameter, as shown in the following example:

```
HelloWorld#1_0
```

In all other instances, the domain/server parameter is required. If it is not specified and WLST finds more than one deployment of the same application on different servers in the domain, you are prompted to specify the domain and the server names.

Web service and web service client applications are deployed directly to WebLogic Server server instances. Each application is managed separately. For example, if the application myapp is deployed to both the AdminServer and server1 instances in the domain mydomain, then you need to issue configuration commands to each of the servers using the appropriate application path name:

```
/mydomain/AdminServer/myapp#1_0
/mydomain/server1/myapp#1_0
```

## Specifying a Service Name

When there are multiple versions (namespaces) of a web service name for Web Service and Web Service clients, you must specify the namespace and the service name using the following format:

```
{http://namespace/}serviceName
```

Note the following:

- For web service and client management commands, and policy management commands, you do not need to enter the namespace if there is only one service name qualified. If there are multiple versions of the service and you do not specify the namespace with the service name, an exception is thrown.

- The namespace ({http://namespace/}) should not be included for a SOA composite.

- For policy set management commands, both the namespace and service name are required for Web Service and Web Service Client (ws-service and ws-client) resource types.

For more information, see "Determining the Namespace for a Web Service" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

# Identifying the Policy Subject

You can navigate to a policy subject in WLST, without having to refer to Fusion Middleware Control. By using the selectWSMPolicySubject command, together with an understanding of the navigation model, you can discover the application, assembly, and subject names by moving down the hierarchy tree. An assembly uniquely identifies a module within an application, for example a .war file.

## Selecting the Application

You can select a specific application for modification if an application name is provided.

If you know only a part of the application name, the argument can be a pattern containing wildcard characters. In this case, all of the applications matching that pattern will be listed. You can then select that application to proceed further. If no argument is provided then all application names will be listed.

### When the application name is known

If you know the name of the application, enter it as the argument to selectWSMPolicySubject command. WLST responds with the names of the assemblies contained in the application.

In the following example, jaxwsejb30ws is entered as the name of the application. WLST responds with #jaxwsejb, the name of the assembly contained in the application.

```
wls:/base_domain/serverConfig> selectWSMPolicySubject('jaxwsejb30ws')

#jaxwsejb

Select any of the assembly name to proceed.
```

### When only a part of the application name is known

If you know only a part of the application name, you can enter a pattern with wildcard characters. In the following example, jax* is entered as the name of the application in the selectWSMPolicySubject command. WLST responds with a list of applications that match the string.

```
wls:/base_domain/serverConfig> selectWSMPolicySubject('jax*')

jaxws_provider
jaxwsejb30ws

Select any of the application name to proceed.

wls:/jrfServer_domain/serverConfig> selectWSMPolicySubject('jaxwsejb30ws')

#jaxws3jb
Select any of the assembly name to proceed
```

### When the application name is not known

If you do not know the name of the application, enter the selectWSMPolicySubject command with no arguments. WLST responds with the names of all applications known to the system. In the following example, the selectWSMPolicySubject

command is entered with no arguments. WLST responds with the names of all applications known to the system.

```
wls:/base_domain/serverConfig> selectWSMPolicySubject()

SimpleRestApp
jaxws_provider
jaxwsejb30ws
wsm-pm

Select any of the application name to proceed.

wls:/jrfServer_domain/serverConfig> selectWSMPolicySubject('jaxwsejb30ws')
#jaxws3jb
Select any of the assembly name to proceed
```

## Selecting the Assembly

You can select a specific assembly for modification if an application name and assembly name is provided.

If you know only a part of the assembly name, the argument can be a pattern containing wildcard characters. In this case, all of the assemblies matching that pattern will be listed. You can then select an assembly to proceed further. If no argument is provided then all assembly names will be listed.

> **Note:** For ws-connection type policy subjects, use an empty string ' ' for the assembly name.

### When the assembly name is known

If you know the name of the assembly, enter it with the application name as arguments to the selectWSMPolicySubject command. WLST responds with the names of the subjects contained in the assembly. In the following example, jaxwsejb30ws is entered as the name of the application and #jaxwsejb is entered as the name of the assembly. WLST responds with a list of all of the subjects contained in the assembly.

```
wls:/base_domain/serverConfig> selectWSMPolicySubject ('jaxwsejb30ws','#jaxwsejb')

WS-SERVICE({http://mycompany.com/targetNamespace}EchoEJBService#EchoEJBServicePort
)
WS-SERVICE({http://mycompany.com/jaxws/tests/concrete}WsdlConcreteService#WsdlConc
retePort)
WS-SERVICE({http://mycompany.com/jaxws/tests}CalculatorService#CalculatorPort)
WS-SERVICE({http://soapinterop.org/DoclitWrapperWTJ}DoclitWrapperWTJService#Doclit
WrapperWTJPort)
WS-SERVICE({http://j2ee.tests.ejb.impl/}JaxwsWithHandlerChainBeanService#JaxwsWith
HandlerChainBeanPort)

Select any of the subject name to proceed.
```

### When only a part of the assembly name is known

If you know only a part of the assembly name, you can enter a pattern with wildcard characters. In the following example, #jaxws* is entered as the partial name of the assembly and jaxwsejb30ws is entered as the name of the application in the

selectWSMPolicySubject command. WLST responds with #jaxwsejb, the name of the assembly contained in the application.

```
wls:/base_domain/serverConfig> selectWSMPolicySubject('jaxwsejb30ws','#jaxws*')

#jaxwsejb
Select any of the assembly name to proceed.

wls:/base_domain/serverConfig> selectWSMPolicySubject ('jaxwsejb30ws','#jaxwsejb')

WS-SERVICE({http://mycompany.com/targetNamespace}EchoEJBService#EchoEJBServicePort
)
WS-SERVICE({http://mycompany.com/jaxws/tests/concrete}WsdlConcreteService#WsdlConc
retePort)
WS-SERVICE({http://mycompany.com/jaxws/tests}CalculatorService#CalculatorPort)
WS-SERVICE({http://soapinterop.org/DoclitWrapperWTJ}DoclitWrapperWTJService#Doclit
WrapperWTJPort)
WS-SERVICE({http://j2ee.tests.ejb.impl/}JaxwsWithHandlerChainBeanService#JaxwsWith
HandlerChainBeanPort)

Select any of the subject name to proceed.
```

### When the assembly name is not known

If you do not know the name of the assembly, enter the name of the application only as an argument to selectWSMPolicySubject. WLST responds with the names of all assemblies known to the system. In the following example, jaxwsejb30ws is entered as the name of the application as an argument in selectWSMPolicySubject command. WLST responds with the names of all assemblies known to the system.

```
wls:/base_domain/serverConfig> selectWSMPolicySubject('jaxwsejb30ws')

#jaxwsejb

Select any of the assembly name to proceed.
```

## Selecting the Subject

You can select a specific policy subject for modification if an application name, assembly name, and policy subject name is provided.

If you know only a part of the policy subject name, the argument can be a pattern containing wildcard characters. In this case, all of the policy subjects matching that pattern will be listed. You can then select a policy subject to proceed further. If no argument is provided then all policy subject names will be listed.

### When the policy subject name is known

If you know the name of the policy subject, enter it with the application name and the assembly name as arguments to the selectWSMPolicySubject command. WLST selects the specified policy subject. In the following example, jaxwsejb30ws is entered as the name of the application, #jaxwsejb is entered as the name of the assembly, and WS-SERVICE({http://mycompany.com/jaxws/tests/concrete}WsdlConcreteService#WsdlConcretePort) is entered as the name of the policy subject. WLST responds that the policy subject has been selected for modification.

```
wls:/base_domain/serverConfig> selectWSMPolicySubject
('jaxwsejb30ws','#jaxwsejb','WS-SERVICE({http://mycompany.com/jaxws/tests/concrete
}WsdlConcreteService#WsdlConcretePort)')
```

```
The policy subject is selected for modification.
```

### When only a part of the policy subject name is known

If you know only a part of the policy subject name, you can enter a pattern with wildcard characters. In the following example, jaxwsejb30ws is entered as the name of the application, #jaxwsejb is entered as the name of the assembly, and ws-service(*) is entered as the name of the policy subject in the selectWSMPolicySubject command. WLST responds with the name of the policy subjects contained in the assembly.

```
wls:/base_domain/serverConfig>  selectWSMPolicySubject
('jaxwsejb30ws','#jaxwsejb', 'ws-service(*)')

WS-SERVICE({http://mycompany.com/targetNamespace}EchoEJBService#EchoEJBServicePort
)
WS-SERVICE({http://mycompany.com/jaxws/tests/concrete}WsdlConcreteService#WsdlConc
retePort)
WS-SERVICE({http://mycompany.com/jaxws/tests}CalculatorService#CalculatorPort)
WS-SERVICE({http://soapinterop.org/DoclitWrapperWTJ}DoclitWrapperWTJService#Doclit
WrapperWTJPort)
WS-SERVICE({http://j2ee.tests.ejb.impl/}JaxwsWithHandlerChainBeanService#JaxwsWith
HandlerChainBeanPort)

Select any of the subject name to proceed.
```

### When the policy subject name is not known

If you do not know the name of the policy subject, enter the name of the application, the name of the assembly as arguments to the selectWSMPolicySubject command. WLST responds with the names of all policy subjects contained in the assembly. In the following example, jaxwsejb30ws is entered as the name of the application, #jaxwsejb as the name of the assembly, and None as the policy subject argument in selectWSMPolicySubject command. WLST responds with the names of all policy subjects contained in the assembly.

```
wls:/base_domain/serverConfig> selectWSMPolicySubject ('jaxwsejb30ws','#jaxwsejb')

WS-SERVICE({http://mycompany.com/targetNamespace}EchoEJBService#EchoEJBServicePort
)
WS-SERVICE({http://mycompany.com/jaxws/tests/concrete}WsdlConcreteService#WsdlConc
retePort)
WS-SERVICE({http://mycompany.com/jaxws/tests}CalculatorService#CalculatorPort)
WS-SERVICE({http://soapinterop.org/DoclitWrapperWTJ}DoclitWrapperWTJService#Doclit
WrapperWTJPort)
WS-SERVICE({http://j2ee.tests.ejb.impl/}JaxwsWithHandlerChainBeanService#JaxwsWith
HandlerChainBeanPort)

Select any of the subject name to proceed.
```

# Web Services WLST Command Categories

Web services WLST commands are divided into the categories described in Table 3–1.

***Table 3–1    Web Services WLST Command Categories***

| Command Category | Definition |
| --- | --- |
| Section , "Session Commands" | Manage a session, which is required by some web service WLST commands, such as those that modify repository documents and policy subject commands, need to be executed in the context of a session. |
| Section , "Policy Subject Commands" | View and manage web service and web service client policy subjects. |
| Section , "Configuration Commands" | View and manage OWSM domain configuration information. |
| Section , "Diagnostic Commands" | Check the status of the WSM components that are required for proper functioning of the product. |
| Section , "Web Service and Client Management Commands" | View and manage web services for the service and client. |
| Section , "Policy Management Commands" | View and manage policy attachment for the service and client. These commands manage both direct policy attachments and global policy attachments in policy sets. |
| Section , "Policy Set Management Commands" | View and manage globally available policy sets within sessions. |
| Section , "OWSM Repository Management Commands" | Manage the OWSM repository with new predefined policies provided in the latest installation of the software, as well as import and export documents into and from the repository. |
| Section , "Token Issuer Trust Configuration Commands" | View and define trusted issuers, trusted distinguished name (DN) lists, and token attribute rule filters for SAML signing certificates. |
| Section , "JKS Keystore Configuration Commands" | View and manage JKS keystore credentials and certificates. |

## Session Commands

Some web service WLST commands, such as those that modify repository documents and policy subject commands, need to be executed in the context of a session. Use the WLST commands listed in Table 3–2 to manage a session.

*Table 3–2    Session Management WLST Commands*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| abortWSMSession | Abort the current modification session, discarding any changes that were made during the session. | Online |
| beginWSMSession | Begin a session to modify a policy subject or the OWSM repository documents. | Online |
| commitWSMSession | Write the contents of the current session to the OWSM repository. | Online |
| describeWSMSession | 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 |

## abortWSMSession

Command Category: Session

Use with WLST: Online

### Description

Aborts the current modification session, discarding any changes that were made during the session. Messages are displayed that describe what was aborted.

An error will be displayed if there is no current session.

### Syntax

```
abortWSMSession()
```

### Examples

The following example aborts the current OWSM session.

```
wls:/wls-domain/serverConfig>abortWSMSession()
```

## beginWSMSession

Command Category: Session

Use with WLST: Online

### Description

Begins a session to modify a policy subject, such as a policy set or a Fusion Middleware web service endpoint. A session can act on a single policy subject only. If a session is already in progress, an error is displayed.

### Syntax

```
beginWSMSession()
```

### Example

The following example begins an OWSM session.

```
wls:/wls-domain/serverConfig>beginWSMSession()
```

## commitWSMSession

Command Category: Session

Use with WLST: Online

### Description

Persists the modifications made within the current session. Messages are displayed that describe what was committed. An error will be displayed if there is no current session.

### Syntax

```
commitWSMSession()
```

### Example

The following example commits the current repository modification session.

```
wls:/wls-domain/serverConfig>commitWSMSession()
```

## describeWSMSession

Command Category: Session

Use with WLST: Online

### Description

Describes the current session. For repository operations, it will either indicate that no actions have been performed in the session, or it will list the name of the document that is being updated, along with the type of update, such as create, modify, or delete. For policy subject operations, it will list the subject identifier.

If there is no current session, the following error is displayed:

```
No actions in session.
```

### Syntax

```
describeWSMSession()
```

### Examples

The following example describes the current session.

```
wls:/wls-domain/serverConfig>describeWSMSession()
```

# Policy Subject Commands

Use the WLST commands listed in Table 3–3 to view and manage web service and web service client policy subjects. For more information about policy subjects, see "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*.

> **Note:** For Java EE web services, no information is displayed. For information about viewing and modifying Java EE web service policy attachments, see Table 3–7.

*Table 3–3    Policy Subject WLST Commands*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| displayWSMEffectivePolicySet | Display the configuration of effective policy set corresponding to a policy subject. | Online |
| listWSMPolicySubjects | List the policy subjects that match the specified application, assembly, and subject patterns. | Online |
| previewWSMEffectivePolicySet | Displays the configuration of an effective policy set corresponding to a policy subject. The display will also include any changes made within the current session when it generates the effective policy set. | Online |
| selectWSMPolicySubject | Select the subject uniquely identified by application, assembly and subject for modification. | Online |

## displayWSMEffectivePolicySet

Command Category: Policy Subject

Use with WLST: Online

> **Note:** This command is valid for Oracle Infrastructure web service and clients only. For Java EE web services, no information is displayed. For information about viewing and modifying Java EE web service policy attachments, see Table 3–7.

### Description

Displays the configuration of the actual runtime policy set and global policy attachment information used at the time of policy enforcement. This policy set and global policy attachment information is stored within the policy subject.

You must start a session and select the policy subject (using `selectWSMPolicySubject`) before initiating the command. If there is no current session and no policy subject selected, an error is displayed.

Compare this command with the "displayWSMPolicySet" on page 3-120 command, which displays only the selected global policy set or the selected local policy set, or with the "previewWSMEffectivePolicySet" on page 3-21, which displays the effective policy set, including changes made to the actual runtime policy set, within the current session.

### Syntax

```
displayWSMEffectivePolicySet()
```

### Examples

The following example for an Oracle Infrastructure web service lists that the policies, `oracle/wss_username_token_service_policy` and `oracle/log_policy`, are in effect at the time of enforcement.

```
wls:/jrfServer_domain/serverConfig> selectWSMPolicySubject('/weblogic/jrfServer_
domain/jaxws-sut','#jaxws-sut-service','WS-SERVICE({http://service.jaxws.wsm.oracl
e/}TestService#TestPort)')

The policy subject is selected for modification.

wls:/jrfServer_domain/serverConfig> displayWSMEffectivePolicySet()

        URI="oracle/http_basic_auth_over_ssl_service_policy", category=security,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true

        The policy subject is secure in this context.
```

## listWSMPolicySubjects

Command Category: Policy Subject

Use with WLST: Online

### Description

Lists the policy subjects that match the specified application, assembly, and subject patterns. You can use the optional detail argument to include effective policy set information in the output. The command does not require starting a session.

For more information about the pattern used to identify a subject, see Section , "Identifying the Policy Subject."

### Syntax

```
listWSMPolicySubjects([application=None],[assembly=None],[subject=None],[detail='f
alse'])
```

| Argument | Definition |
|---|---|
| *application* | Optional. Pattern identifying applications. |
| *assembly* | Optional. Pattern identifying assemblies. |
| *subject* | Optional. Pattern identifying subjects. |
| *detail* | Optional. Specifies whether to include effective policy set information in the output. The default value is false. |
| | For each directly attached policy, the local.policy.reference.source configuration property is provided identifying the source of the attachment. For more information, see "Determining the Source of Policy Attachments" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*. |

To simplify searching for a particular subject, the *application*, *assembly*, or *subject* argument can specify a pattern containing the wildcard character (*). In this case, all the subjects matching that pattern will be listed.

### Examples

The following invocation of the listWSMPolicySubjects command with detail='true' returns the application, assembly, and subject information for all subjects being managed in the entire domain

Note that the local.policy.reference.source configuration property is provided for the directly attached policy identifying its source as LOCAL_ATTACHMENT, indicating that it was attached using either Fusion Middleware Control or WLST. For more information about the local.policy.reference.source configuration property and a list of valid values, see "Determining the Source of Policy Attachments" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

```
wls:/base_domain/serverConfig> listWSMPolicySubjects(detail='true')
Application: /weblogic/base_domain/jaxwsejb30ws
  Assembly: #jaxwsejb
    Subject:
WS-SERVICE({http://mycompany.com/targetNamespace}EchoEJBService#EchoEJBServicePort
)
```

```
Context : no constraint
        URI="oracle/wss_username_token_service_policy", category=security,
policy-status=enabled; source=global policy set "username", scope="DOMAIN('*')";
reference-status=enabled; effective=true
        URI="oracle/mex_request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="local.policy.reference.source", value="IMPLIED_
FEATURE"
        URI="oracle/mtom_encode_fault_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="local.policy.reference.source", value="IMPLIED_
FEATURE"
        URI="oracle/max_request_size_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="local.policy.reference.source", value="IMPLIED_
FEATURE"
                Property name="max.request.size", value="-1"
        URI="oracle/request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="local.policy.reference.source", value="IMPLIED_
FEATURE"
        URI="oracle/soap_request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="local.policy.reference.source", value="IMPLIED_
FEATURE"
        URI="oracle/ws_logging_level_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="logging.level", value=""
                Property name="local.policy.reference.source", value="IMPLIED_
FEATURE"
        URI="oracle/test_page_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="local.policy.reference.source", value="IMPLIED_
FEATURE"
        URI="oracle/wsdl_request_processing_service_policy", category=wsconfig,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                Property name="local.policy.reference.source", value="IMPLIED_
FEATURE"

        The policy subject is secure in this context.

...
```

Invoking the `listWSMPolicySubjects` command with `('jax*')` as the argument returns all subjects in applications that begin with `jax`; in our example, all subjects belonging to the `jaxwsejb30ws` application:

```
wls:/base_domain/serverConfig> listWSMPolicySubjects('jax*')

Application: /weblogic/base_domain/jaxwsejb30ws
```

```
      Assembly: #jaxwsejb
         Subject:
WS-SERVICE({http://mycompany.com/targetNamespace}EchoEJBService#EchoEJBServicePort
)

         Subject:
WS-SERVICE({http://mycompany.com/jaxws/tests/concrete}WsdlConcreteService#WsdlConc
retePort)

         Subject:
WS-SERVICE({http://mycompany.com/jaxws/tests}CalculatorService#CalculatorPort)
```

The following command returns all RESTful resource subjects in all applications. If there are no RESTful resources in an application, the following message is returned:

```
Subject: No matching subject found for "REST*"
```

```
wls:/base_domain/serverConfig> listWSMPolicySubjects(subject='REST*')

Application: /weblogic/base_domain/jaxrs_pack1

  Assembly: #jaxrs_pack1.war

    Subject: REST-Resource(Jersey)


Application: /weblogic/base_domain/jaxwsejb30ws

  Assembly: #jaxwsejb

    Subject: No matching subject found for "REST*".


Application: /weblogic/base_domain/soa-infra

  Assembly: #integration/services/RuntimeConfigService

    Subject: REST-Resource(oracle.bpm.rest.webapp.BPMApplication)
```

## previewWSMEffectivePolicySet

Command Category: Policy Subject

Use with WLST: Online

### Description

Displays the configuration of the effective policy set corresponding to the policy subject. The display will also include any changes made within current session when it generates the effective policy set.

You must start a session and select the policy subject (using `selectWSMPolicySubject`) before initiating the command. An error will display if no policy subject is selected.

See also "displayWSMEffectivePolicySet" on page 3-17, which displays the actual policy set used at the time of enforcement, but does not display any changes made to the policy set during the current session.

### Syntax

```
previewWSMEffectivePolicySet()
```

### Examples

```
wls:/wls-domain/serverConfig>previewWSMEffectivePolicySet()
```

## selectWSMPolicySubject

Command Category: Policy Subject

Use with WLST: Online

### Description

Within a session, selects a policy subject for modification. You uniquely specify a policy subject by the application, assembly, and policy subject name. Once selected, the policy management commands can be used to modify the directly attached policy set for the policy subject.

You must start a session (beginWSMSession) before performing any policy management edits or policy set transactions. You must also select the policy subject that you want to modify before issuing policy management commands. 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 "Identifying the Policy Subject" on page 3-6 and "Identifying and Selecting the Policy Subject Using WLST" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

### Syntax

```
selectWSMPolicySubject([application=None],[assembly=None],[subject=None])
```

| Argument | Description |
|----------|-------------|
| application | Name of the application. |
| assembly | Name of the assembly. Uniquely identifies the module within an application. |
| subject | Name of the policy subject. |

> **Note:**   Any of the three arguments can specify a pattern containing wildcard character "*". In this case, all the names matching that pattern will be listed. You need to select the name uniquely identifying the subject. The pattern can be specified only for the last unknown entity.

### Examples

The following example selects the TestService#TestPort port in the jaxws-sut-service module (assembly) that belongs to the jaxws-sut application.

```
wls:/jrfServer_domain/serverConfig> selectWSMPolicySubject('/weblogic/jrfServer_
domain/jaxws-sut','#jaxws-sut-service','WS-SERVICE({http://service.jaxws.wsm.oracl
e/}TestService#TestPort)')

The policy subject is selected for modification.
```

The following example selects the jersey RESTful resource in the #restservice module (assembly) that belongs to the helloworld application.

```
wls:/jrfServer_domain/serverConfig>
selectWSMPolicySubject('helloworld','#restservice','REST-Resource(Jersey)')
```

```
The policy subject is selected for modification.
```

See "Identifying the Policy Subject" on page 3-6 for additional examples.

# Configuration Commands

Use the WLST commands listed in Table 3–4 to view and configure the OWSM domain.

> **Note:** The setConfiguration command has been deprecated. It is recommended that you use the setWSMConfiguration command described in "setWSMConfiguration" on page 3-28.

*Table 3–4    OWSM Environment WLST Commands*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| configureWSMKeystore | Set the keystore configuration properties. | Online |
| displayWSMConfiguration | Display the full configuration properties and their values and groups for the specified product. | Online |
| setWSMConfiguration | Set the configuration properties of the specified product. | Online |

## configureWSMKeystore

Command Category: Configuration

Use with WLST: Online

### Description

Sets the configuration properties for the OWSM keystore.

For more information, see "Configuring the OWSM Keystore Using WLST" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

---

> **Note:** Changes to the keystore configuration at the domain level require that you restart the server.

---

### Syntax

```
configureWSMKeystore(context, keystoreType, location, keystorePassword, signAlias,
signAliasPassword, cryptAlias, cryptAliasPassword)
```

| Arguments | Description |
|---|---|
| *context* | Optional. The context of the configuration document in which the modifications will be done. |
| *keystoreType* | Optional. The keystore type category of the property. Valid keystore types are JKS, KSS, PKCS11, and LUNA. |
| *location* | Optional. For JKS, it is the absolute location of the keystore or location relative to the fmwconfig directory. For KSS, the format of location should be kss://stripeName/keystoreName The default is kss://owsm/keystore. |
| *keystorePassword* | Optional. The keystore password of the keystore configured. It is required for JKS and PKCS11. |
| *signAlias* | Optional. The Alias of the sign key. It is required for JKS and PKCS11. |
| *signAliasPassword* | Optional. Password of the Alias of the sign key. It is required for JKS and PKCS11. |
| *cryptAlias* | Optional. The Alias of the Encryption key. It is required for JKS and PKCS11. |
| *cryptAliasPassword* | Optional. Password of the Alias of the Encryption key. It is required for JKS and PKCS11. |

### Examples

The following example configures the JKS keystore default-keystore.jks in the domain myDomain. It provides the keystore password oratest123, the sign alias oraAlias, the sign alias password ora234, the encryption alias oraCryptAlias, the encryption alias password ora123.

```
wls:/jrfServer_domain/serverConfig> configureWSMKeystore ('/WLS/myDomain','JKS',
'./default-keystore.jks','oratest123', 'oraAlias','ora234','oraCryptAlias',
'ora123')
```

The following example configures the KSS keystore at `kss://owsm/keystore` in the domain `myDomain`. It provides the sign alias `oraAlias`, and the encryption alias `oraCryptAlias`.

```
 wls:/jrfServer_domain/serverConfig> configureWSMKeystore
('/WLS/myDomain',keystoreType='KSS', location='kss://owsm/keystore',
signAlias='oraAlias', cryptAlias='encAlias')
```

## displayWSMConfiguration

Command Category: Configuration

Use with WLST: Online

### Description

Displays the full set of configuration properties, and their values and groups, for the product specified in the context. If a property is not defined in the configuration document associated with the context, then the default value defined for the product is displayed. If a context is not specified, then the set of properties matching the current context is displayed.

For more information, see "Managing OWSM Domain Configuration Using WLST" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

### Syntax

```
displayWSMConfiguration([context=None])
```

| Arguments | Description |
| --- | --- |
| *context* | Optional. The context of the configuration document from which property values are displayed. If a *context* is not specified, then the set of properties matching the current context is displayed. |
| | To display the default set of properties along with their values, use "/" as the context value." |

### Examples

The following example displays the configuration contained in the configuration document in the repository.

```
wls:/jrfServer_domain/serverConfig> displayWSMConfiguration()
```

The following example displays the configuration for the base_domain domain.

```
wls:/jrfServer_domain/serverConfig> displayWSMConfiguration('WLS/base_domain')
```

## setWSMConfiguration

Command Category: Configuration

Use with WLST: Online

### Description

Sets the configuration properties of a domain. The properties are stored in a configuration document for the domain. If a configuration document does not exist, a new one is created.

A new property with values and/or groups of values can be added inside the configuration document. The set of acceptable properties is determined from the default set of properties supported by the product. Specific property values or groups of values can be removed from the configuration document. The configuration document itself is removed if no properties exist in it.

For more information, see "Managing OWSM Domain Configuration Using WLST" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

### Syntax

```
setWSMConfiguration(context, category, name, [group=None], [values=None])
```

| Arguments | Description |
| --- | --- |
| *context* | Optional. The context of the configuration document to be modified. If a context is not provided or is set to None, then the configuration document associated with the currently connected domain is used. For example /WLS/base_domain. |
| *category* | The category of the property. This is verified against the default set of properties to ensure it is acceptable for the context.<br><br>Use the displayWSMConfiguration command to see the category name associated with each property. |
| *name* | The name of the property. This is verified against the default set of properties to ensure it is acceptable for the context. |
| *group* | Optional. A group containing the set of values to add in a configuration document. If the group exists, and this value is set to None, the group is removed. |
| *values* | Optional. The array of values to set for a property or group inside the configuration document. |

### Examples

The following example resets the entire configuration for the domain myDomain to its default values.

```
wls:/jrfServer_domain/serverConfig> setWSMConfiguration('/WLS/myDomain')
```

The following command resets the value of the clock.skew property in myDomain to 500.

```
wls:/jrfServer_domain/serverConfig>
setWSMConfiguration('/WLS/myDomain','Agent','clock.skew',None, ['500'])
```

The following command resets the value of the clock.skew property in myDomain to its default value.

```
wls:/jrfServer_domain/serverConfig>
setWSMConfiguration('/WLS/myDomain','Agent','clock.skew',None,None)
```

# Diagnostic Commands

Use the WLST command in this section to check the status of the WSM components that are required for proper functioning of the product.

## checkWSMStatus

Command Category: Diagnostic

Use with WLST: Online

### Description

Checks the status of the OWSM components that are required for proper functioning of the product. The OWSM 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: <br><br> ■ wsm-pm—Policy Manager. Checks the configuration state of the policy manager component. <br><br> ■ 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. <br><br> ■ 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 wsm-pm application. This value is required for checking enforcement through an agent component, for example, <br><br> `checkWSMStatus('agent', 'http://localhost:7001')` <br><br> 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) will be 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:port/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
```

# Web Service and Client Management Commands

Use the WLST commands listed in Table 3–5 to view and manage web services for deployed, active, and running web service applications.

---

**Note:**   The commands listed in Table 3–5 have an `application` argument.

In an multi-tenant environment, if you intend to target a specific application instance within a tenant's partition, then you must include the partition name as part of the application as follows:

`/domain/server/application#version$partition`

However, if you are targeting a domain-scoped application, then you do not have to include the partition name. You can use the `application` argument as follows:

`/domain/server/application#version`

---

*Table 3–5    Web Service and Client Management WLST Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| listWebServiceClientPorts | List web service client ports information for an application or SOA composite. | Online |
| listWebServiceClients | List web service client information for an application, SOA composite, or domain. | Online |
| listWebServiceClientStubProperties | List web service client port stub properties for an application or SOA composite. | Online |
| listWebServicePorts | List the web service ports for a web service application or SOA composite. | Online |
| listWebServices | List the web service information for an application, composite, or domain. | Online |
| setWebServiceClientStubProperties | Configure the set of stub properties of a web service client port for an application or SOA composite. | Online |
| setWebServiceClientStubProperty | Set, change, or delete a single stub property of a web service client port for an application or SOA composite. | Online |

## listWebServiceClientPorts

Command Category: Web Service and Client Management

Use with WLST: Online

### Description

Lists the web service port names and the endpoint URLs for web service clients in an application or SOA composite.

The output will display the name of the web service client/reference port. For example:

```
AppModuleServiceSoapHttpPort
```

### Syntax

```
listWebServiceClientPorts(application,moduleOrCompName,moduleType,serviceRefName)
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to list the web services port information. For example, `/domain/server/application#version_number` |
| | To list the client port information for an application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to list the web service client port information. |
| | To list the client port information for a SOA composite, the composite name is required (for example, `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`. |
| *moduleType* | Module type. Valid options are: |
| | ■ `soa`—SOA composite. |
| | ■ `web`—Oracle Infrastructure web services packaged as a Web module (including an EJB). |
| | ■ `wls`—Java EE web services. |
| | ■ `wsconn`—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy. |
| *serviceRefName* | Service reference name of the application or SOA composite for which you want to list the web service client port information. |
| | When the client is an asynchronous web service callback client, the `serviceRefName` argument must be set to `callback`. |

### Examples

The following example lists the client ports for the `WssUsernameClient` Web module in the `/base_domain/server1/jwsclient_1#1.1.0` application. Note that the `moduleType` is set to `wsconn`, and the `serviceRefName` is set to `WssUsernameClient`.

```
wls:/base_domain/serverConfig> listWebServiceClientPorts
('/base_domain/server1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient')
```

The following example lists the client ports in the `default/HelloWorld[1.0]` SOA composite. Note that the `moduleType` is set to `soa`, and the `serviceRefName` is set to `client`.

```
wls:/base_domain/serverConfig> listWebServiceClientPorts(None,
'default/HelloWorld[1.0]','soa','client')
```

## listWebServiceClients

Command Category: Web Service and Client Management

Use with WLST: Online

## Description

Lists web service clients information for an application, SOA composite, or domain. If neither an application nor a composite is specified, the command lists information about all Web service clients in all applications and composites for every server instance in the domain. If an application is not specified, the command lists information about all web service clients in all applications for every server instance in the domain.

You can specify the amount of information to be displayed in the output using the `detail` argument. When specified, the output provides endpoint (port) and policy details for clients in the domain, the secure status of the endpoints, any configuration overrides and constraints, and if the endpoints have a valid configuration. A subject is considered secure if the policies attached to it (either directly or globally) enforce authentication, authorization, or message protection behaviors. Because you can specify the priority of a global or directly attached policy (using the `reference.priority` configuration override), the `effective` field indicates if the directly attached policies are in effect for the endpoint.

The `local.policy.reference.source` configuration property is provided for each directly attached policy identifying the source of the attachment. For more information about the `local.policy.reference.source` configuration property and a list of valid values, see "Determining the Source of Policy Attachments" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

Note that to simplify endpoint management, all directly attached policies are shown in the output regardless of whether they are in effect. In contrast, only globally attached policies that are in effect for the endpoint are displayed. For more information, see "How the Effective Set of Policies is Calculated" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

The output is listed by each application deployed as shown in the following examples:

This example shows the output of an *unsecured* endpoint:

```
wls:/jrfServer_domain/serverConfig> listWebServiceClients(detail=true)

/jrfServer_domain/jrfServer_admin/ADFDCDecoupling_Project1_ADFDCDecoupling :
        moduleName=testadfbc, moduleType=wsconn, serviceRefName=AppModuleService
                AppModuleServiceSoapHttpPort

                The policy subject is not secure in this context.

/soa_domain/soa_server1/soa-infra :
        compositeName=default/Basic_SOA_Client[1.0], moduleType=soa,
serviceRefName=Service1
                Basic_soa_service_pt
serviceWSDLURI=http://host.example.com:1234/soa-infra/services/default/Basic_SOA_
service/Basic_soa_service.wsdl
                oracle.webservices.contentTransferEncoding=base64
                oracle.webservices.charsetEncoding=UTF-8
                oracle.webservices.operationStyleProperty=document
                wsat.flowOption=WSDLDriven
                oracle.webservices.soapVersion=soap1.1
```

```
                         oracle.webservices.chunkSize=4096
                         oracle.webservices.session.maintain=false
                         oracle.webservices.preemptiveBasicAuth=false

oracle.webservices.encodingStyleProperty=http://schemas.xmlsoap.org/soap/encoding/
                         oracle.webservices.donotChunk=true
                         No attached policies found; endpoint is not secure.
```

This example shows the output for a *secured* endpoint. Note that the
`local.policy.reference.source` configuration property is provided for the directly
attached policy identifying its source as `LOCAL_ATTACHMENT`, indicating that it was
attached using either Fusion Middleware Control or WLST. For more information
about the `local.policy.reference.source` configuration property and a list of valid
values, see "Determining the Source of Policy Attachments" in *Securing Web Services
and Managing Policies with Oracle Web Services Manager*.

```
wls:/jrfServer_domain/serverConfig> listWebServiceClients(detail=true)

/jrfServer_domain/jrfServer_admin/ADFDCDecoupling_Project1_ADFDCDecoupling :
        moduleName=testadfbc, moduleType=wsconn, serviceRefName=AppModuleService
                AppModuleServiceSoapHttpPort
serviceWSDLURI=http://host.example.com:1234/ADFBCDecoupling-ADFBCDecoupling-contex
t-root/AppModuleService?wsdl
                URI="oracle/wss10_saml_token_with_message_protection_client_
policy", category=security, policy-status=enabled; source=local policy set;
reference-status=enabled; effective=true
                        Property name="local.policy.reference.source", value="LO
CAL_ATTACHMENT"

        The policy subject is secure in this context.
```

## Syntax

```
listWebServiceClients(application,composite,[detail])
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to list the web service clients. For example, `/domain/server/application#version_number` |
| | If specified, all web services clients in the application are listed. |
| *composite* | Name of the SOA composite for which you want to list the Web service clients. For example, `default/HelloWorld[1.0]` |
| | If specified, all Web service clients in the composite are listed. |
| *detail* | Optional. Specifies whether to list port and policy details for the web service clients. |
| | For each directly attached policy, the `local.policy.reference.source` configuration property is provided identifying the source of the attachment. For more information, see "Determining the Source of Policy Attachments" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*. |
| | Valid values are: |
| | ■ `true`—Output includes details about the clients, ports, policies, and whether the endpoint is secure or not. |
| | ■ `false`—Output lists only the clients. The default is false. |

## Examples

The following example lists information for all web service clients in the domain.

```
wls:/wls-domain/serverConfig>listWebServiceClients()
```

The following example lists the web service clients for the application jwsclient_
1#1.10 for the server server1 in the domain base_domain.

```
wls:/wls-domain/serverConfig>listWebServiceClients('base_domain/server1/jwsclient_
1#1.10')
```

The following example lists the Web service clients for the SOA composite
default/HelloWorld[1.0].

```
wls:/wls-domain/serverConfig>listWebServiceClients(None,'default/HelloWorld[1.0]')
```

The following example lists details for all of the web service clients in the domain.

```
wls:/wls-domain/serverConfig>listWebServiceClients(None,None,true)
```

## listWebServiceClientStubProperties

> **Note:** This command applies to Oracle Infrastructure web service clients only.

Command Category: Web Service and Client Management

Use with WLST: Online

### Description

Lists web service client port stub properties for an application or SOA composite.

### Syntax

```
listWebServiceClientStubProperties(application, moduleOrCompName, moduleType,
serviceRefName, portInfoName)
```

| Argument | Definition |
|----------|------------|
| *application* | Name and path of the application for which you want to list the web services client port stub properties. For example, `/domain/server/application#version_number`<br><br>To list the client port stub properties information for an application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to list the web services client port stub properties.<br><br>To list the client port stub properties information for a SOA composite, the composite name is required (for example, `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`. |
| *moduleType* | Module type. Valid options are:<br><br>■ `soa`—SOA composite.<br><br>■ `web`—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br><br>■ `wsconn`—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy. |
| *serviceRefName* | Service reference name of the application or SOA composite for which you want to list the web service client port stub properties. |
| *portInfoName* | The name of the client port for which you want to list the stub properties. |

### Example

The following example lists the client port stub properties for the `JRFWssUsernamePort` port of the `WssUsernameClient` Web module in the `/base_domain/server1/jwsclient_1#1.1.0` application. Note that the `moduleType` is set to `wsconn`, and the `serviceRefName` is set to `WssUsernameClient`.

```
wls:/base_domain/serverConfig>listWebServiceClientStubProperties
('/base_domain/server1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort')
```

## listWebServicePorts

Command Category: Web Service and Client Management

Use with WLST: Online

### Description

Lists the web service port names and the endpoint URLs for a web service application or SOA composite.

The output will display the port name and endpoint URL of the web service port. For example:

```
JRFWssUsernamePort        http://localhost:7001/j2wbasicPolicy/WssUsername
```

### Syntax

```
listWebServicePorts(application,moduleOrCompName,moduleType,serviceName)
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to list the web services port information. For example, /domain/server/application#version_number |
| | To list the port information for an application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to list the web services port information. |
| | To list the port information for a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |
| *moduleType* | Module type. Valid options are: |
| | ■ soa—SOA composite. |
| | ■ web—Oracle Infrastructure web services packaged as a Web module (including an EJB). |
| | ■ wls—Java EE web services. |
| *serviceName* | Name of the web service in the application or SOA composite for which you want to list the port information. For example, {http://namespace/}serviceName. Note that the namespace ({http://namespace/}) should not be included for a SOA composite. |

### Example

The following example lists the web service ports and endpoint URLs for the Oracle Infrastructure web service j2wbasicPolicy service in the base_domain/AdminServer/HelloWorld#1_0 application. Note that the WssUsernameService module name is specified, and the moduleType is set to web.

```
wls:/base_domain/serverConfig> listWebServicePorts
( '/base_domain/AdminServer/HelloWorld#1_0',
'WssUsernameService','web','{http://namespace/}j2wbasicPolicy')

JRFWssUsernamePort        http://localhost:7001/j2wbasicPolicy/WssUsername
```

The following example lists the web service ports and endpoint URLs for the Java EE web service `helloWorldJaxws` in the `wls-domain/AdminServer/helloWorldJaxws` application. Note that the `moduleType` is set to `wls`.

```
wls:/wls-domain/serverConfig> listWebServicePorts
('/wls-domain/AdminServer/helloWorldJaxws','helloWorldJaxws#1!helloWorldJaxws',
'wls','helloWorldJaxws')

helloWorldJaxwsSoapHttpPort
```

## listWebServices

Command Category: Web Service and Client Management

Use with WLST: Online

### Description

Lists the web service information for an application, SOA composite, or domain. If you do not specify a web service application or a SOA composite, the command lists all services in all applications and composites for every server instance in the domain.

You can specify the amount of information to be displayed in the output using the `detail` argument. When enabled, the output provides endpoint (port) and policy details for all applications and composites in the domain, the secure status of the endpoints, any configuration overrides and constraints, and if the endpoints have a valid configuration. In addition, the `local.policy.reference.source` configuration property is provided for each directly attached policy identifying the source of the attachment, as described in "Determining the Source of Policy Attachments" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

A subject is considered secure if the policies attached to it (either directly or globally) enforce authentication, authorization, or message protection behaviors. Because you can specify the priority of a global or directly attached policy (using the `reference.priority` configuration override), the `effective` field indicates if the directly attached policies are in effect for the endpoint.

Note that to simplify endpoint management, all directly attached policies are shown in the output regardless of whether they are in effect. In contrast, only globally attached policies that are in effect for the endpoint are displayed. For more information, see "How the Effective Set of Policies is Calculated" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

The output is listed by each application deployed as shown in the following example:

```
/domain/server/application#version_number:
     moduleName=helloModule, moduleType=web,
serviceName={http://namespace/}service
/base_domain/AdminServer/soa-infra:
     compositeName=default/HelloWorld[1.0],  moduleType=soa, serviceName=service
```

> **Notes:** The `listWebServices` command output does not include details on SOA components, including policy attachments.
>
> For applications assembled prior to 11*g* Release 1, (11.1.1.6), the namespace is not displayed with the `serviceName` in the output.

### Syntax

```
listWebServices (application,composite,[detail])
```

| Argument | Definition |
| --- | --- |
| *application* | Name and path of the application for which you want to list the web services. For example, `/domain/server/application#version_number` |
| | If specified, all web services in the application are listed. |

| Argument | Definition |
|----------|------------|
| *composite* | Name of the SOA composite for which you want to list the Web services. For example, default/HelloWorld[1.0] |
| | If specified, all Web services in the composite are listed. |
| *detail* | Optional. Specifies whether to list port and policy details for the web service. |
| | For each directly attached policy, the local.policy.reference.source configuration property is provided identifying the source of the attachment. For more information, see "Determining the Source of Policy Attachments" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*. |
| | Valid values are: |
| | ■  true—Output includes details about the service, the port, and the policies. |
| | ■  false—Output lists only the services. The default is false. |

## Examples

The following example for an Oracle Infrastructure web service lists all the web services in all applications and composites in the domain. Sample output is shown in this example.

```
wls:/base_domain/serverConfig> listWebServices()

/base_domain/AdminServer/soa-infra :
     compositeName=default/HelloWorld[1.0], moduleType=soa, serviceName=service
     compositeName=default/Project1[1.0], moduleType=soa,
serviceName=bpelprocess1_client_ep
/base_domain/AdminServer/jaxwsejb30ws :
        moduleName=jaxwsejb, moduleType=web,
serviceName=JaxwsWithHandlerChainBeanService
        moduleName=jaxwsejb, moduleType=web, serviceName=WsdlConcreteService
        moduleName=jaxwsejb, moduleType=web, serviceName=EchoEJBService
        moduleName=jaxwsejb, moduleType=web, serviceName=CalculatorService
        moduleName=jaxwsejb, moduleType=web, serviceName=DoclitWrapperWTJService
```

The following example for an Oracle Infrastructure web service sets the detail argument to true. Sample output is shown in this example. Security policies are shown in bold text.

Note that the reference priority of the globally attached policy is set to 10 and the directly attached policy is not in effect for the endpoint CalculatorPort in the application jaxwsejb30ws.

Also, note that the local.policy.reference.source configuration property is provided for each directly attached policy identifying the source of the attachment. For more information about the local.policy.reference.source configuration property and a list of valid values, see "Determining the Source of Policy Attachments" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

```
wls:/base_domain/serverConfig> listWebServices(detail='true')

/base_domain/AdminServer/jaxwsejb30ws :
moduleName=jaxwsejb, moduleType=web, serviceName=CalculatorService
          CalculatorPort  http://host.example.com:1234/jaxwsejb/Calculator
          URI="oracle/wss10_saml20_token_with_message_protection_service_
policy",
category=security, policy-status=enabled; source=global policy set "
```

```
                MyPolicySet1", scope="DOMAIN('*')"; reference-status=enabled; effective=true
                              Property name="reference.priority", value="10"
              URI="oracle/mex_request_processing_service_policy",
 category=wsconfig, policy-status=enabled; source=local policy set;
 reference-status=enabled; effective=true
                              Property name="local.policy.reference.source", value="IM
PLIED_FEATURE"
              URI="oracle/mtom_encode_fault_service_policy", category=wsconfig,
 policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                              Property name="local.policy.reference.source", value="IM
PLIED_FEATURE"
              URI="oracle/max_request_size_policy", category=wsconfig,
 policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                              Property name="local.policy.reference.source", value="IM
PLIED_FEATURE"
                              Property name="max.request.size", value="-1"
              URI="oracle/request_processing_service_policy", category=wsconfig,
 policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                              Property name="local.policy.reference.source", value="IM
PLIED_FEATURE"
              URI="oracle/soap_request_processing_service_policy",
category=wsconfig,
 policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                              Property name="local.policy.reference.source", value="IM
PLIED_FEATURE"
              URI="oracle/ws_logging_level_policy", category=wsconfig,
 policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                              Property name="logging.level", value=""
                              Property name="local.policy.reference.source", value="IM
PLIED_FEATURE"
              URI="oracle/test_page_processing_service_policy", category=wsconfig,
 policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                              Property name="local.policy.reference.source", value="IM
PLIED_FEATURE"
              URI="oracle/wsdl_request_processing_service_policy",
category=wsconfig,
 policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
                              Property name="local.policy.reference.source", value="IM
PLIED_FEATURE"
              URI="oracle/http_saml20_token_bearer_service_policy",
category=security,
 policy-status=enabled; source=local policy set; reference-status=enabled;
reference-status=enabled; effective=false
                              Property name="local.policy.reference.source",
value="ANNOTATION"

        The policy subject is secure in this context.
```

The following example for a Java EE web service sets the detail argument to true. Sample output is shown in this example. The output lists all the web services in all applications and composites in the domain.

```
/base_domain/AdminServer/SimpleJAXWS :
   moduleName=SimpleJAXWS#1!SimpleEjbService, moduleType=wls,
serviceName=SimpleEjbService
      SimplePort
      URI="oracle/http_basic_auth_over_ssl_service_policy", category=security,
policy-status=enabled; source=local policy set; reference-status=enabled;
effective=true
            Property name="local.policy.reference.source", value="LOCAL_
ATTACHMENT"

        The policy subject is secure in this context.

        moduleName=SimpleJAXWS#1!SimpleImplService, moduleType=wls,
serviceName=SimpleImplService
            SimplePort
                has Operation level ws-policy
            Attached policy or policies are valid; endpoint is not secure.
```

## setWebServiceClientStubProperties

> **Note:** This command applies to Oracle Infrastructure web service clients only.

Command Category: Web Service and Client Management

Use with WLST: Online

### Description

Configures the set of stub properties of a web service client port for an application or SOA composite.

This command configures or resets all of the stub properties for the OWSM client security policy attached to the client. Each property that you list in the command is set to the value you specify. If a property that was previously set is not explicitly specified in this command, it is reset to the default for the property. If no default exists, the property is removed.

### Syntax

```
setWebServiceClientStubProperties(application, moduleOrCompName, moduleType,
serviceRefName, portInfoName, properties)
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to reset the web services client port stub properties. For example, `/domain/server/application#version_number` |
| | To configure or reset the client port stub properties for an application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to reset the web services client port stub properties. |
| | To configure or reset client port stub properties for a SOA composite, the composite name is required (for example, `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`. |
| *moduleType* | Module type. Valid options are:<br><br>■ `soa`—SOA composite.<br><br>■ `web`—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br><br>■ `wsconn`—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy. |
| *serviceRefName* | Service reference name of the application or SOA composite for which you want to reset the web service client port stub properties. |
| *portInfoName* | The name of the client port for which you want to reset the stub properties. |

| Argument | Definition |
| --- | --- |
| *properties* | The list of properties to be set or changed. Properties must be specified using the following format: |
| | `("property","value")` |
| | For example: |
| | `[("keystore.recipient.alias","oracle"),`<br>`("csf-key","oracle")]` |
| | To remove a property or clear the value assigned to it, specify a blank `""` value. For example: |
| | `[("csf-key","")]` |
| | To remove all the properties of the client port, set this argument to `None`. |
| | Sample client port stub properties are as follows: |
| | ■ `oracle.webservices.auth.username` |
| | ■ `oracle.webservices.auth.password` |
| | ■ `keystore.recipient.alias` |
| | ■ `csf-key` |
| | ■ `saml.issuer.name` |
| | ■ `javax.xml.ws.session.maintain` |
| | ■ `wsat.Version` —SOA references only |
| | ■ `wsat.flowOption`—SOA references only |

### Example

The following example resets the client port stub properties `ROLE` and `keystore.recipient.alias` to `ADMIN` and `orakey`, respectively. Any other properties that were previously set for this client port are either reset to the default or removed. The client port is `JRFWssUsernamePort` of the `WssUsernameClient` Web module in the `/base_domain/server1/jwsclient_1#1.1.0` application. Note that the `moduleType` is set to `wsconn`, and the `serviceRefName` is set to `WssUsernameClient`.

```
wls:/base_domain/serverConfig>setWebServiceClientStubProperties('/base_
domain/server1/jwsclient_1#1.1.0',
'WssUsernameClient','wsconn','WssUsernameClient','JRFWssUsernamePort',
[("ROLE","ADMIN"),("keystore.recipient.alias","orakey")] )
```

## setWebServiceClientStubProperty

Command Category: Web Service and Client Management

Use with WLST: Online

### Description

Sets, changes, or deletes a single stub property of a web service client port for an application or SOA composite.

### Syntax

```
setWebServiceClientStubProperty(application, moduleOrCompName, moduleType,
serviceRefName,portInfoName,propName,[propValue])
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to set the web services client port stub property. For example, /domain/server/application#version_number |
| | To set a client port stub property for an application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to set the web services client port stub property. |
| | To set a client port stub property for a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |
| *moduleType* | Module type. Valid options are: |
| | ■ soa—SOA composite. |
| | ■ web—Oracle Infrastructure web services packaged as a Web module (including an EJB). |
| | ■ wls—Java EE web services. |
| | ■ wsconn—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy. |
| *serviceRefName* | Service reference name of the application or SOA composite for which you want to set the web service client port stub property. |
| *portInfoName* | The name of the client port for which you want to set the stub property. |
| *propName* | Stub property name that you want to set, change, or delete. For example, 'keystore.recipient.alias'. |
| *propValue* | Optional. The stub property value, for example, 'orakey'. |
| | To remove the property, specify a blank "" value. |

### Example

The following example sets the client port stub property keystore.recipient.alias to the value orakey for the client port JRFWssUsernamePort. The port is a client port of the WssUsernameClient Web module in the /base_domain/server1/jwsclient_1#1.1.0 application. Note that the moduleType is set to wsconn, and the serviceRefName is set to WssUsernameClient.

```
wls:/base_domain/serverConfig>setWebServiceClientStubProperty
('/base_domain/server1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort','keystore.recipient.alias','orakey')
```

## Policy Management Commands

> **Note:** The policy management commands for Java EE Web Services (or clients) listed in Table 3–7 have been deprecated in this release for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure web services, to manage OWSM directly attached policies in release 12*c*, it is recommended that you use the new WLST commands listed in Table 3–6. For a complete list of deprecated commands, see "Deprecated Commands for Oracle Infrastructure Web Services" in *Release Notes for Oracle Fusion Middleware Infrastructure*.

Use the WLST commands listed in Table 3–6 to manage Oracle Infrastructure and RESTful Web Services direct and global policy attachments.

***Table 3–6    Oracle Infrastructure and RESTful Web Services and Clients - WLST Commands for Direct Policy Attachments***

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| attachWSMPolicy | Attach a policy to the selected policy subject or policy set document within a session. | Online |
| attachWSMPolicies | Attach multiple policies to the selected policy subject or policy set document within a session. | Online |
| detachWSMPolicy | Detach a policy from the selected policy subject or policy set document within a session. | Online |
| detachWSMPolicies | Detach multiple policies from the selected policy subject or policy set document within a session. | Online |
| enableWSMPolicies | Enable or disable multiple policies that are attached to the selected policy subject or policy set document within a session. | Online |
| enableWSMPolicy | Enable or disable a policy that is attached to the selected policy subject or policy set document within a session. | Online |
| listAvailableWebServicePolicies | Display a list of all the available OWSM policies by category or subject type. | Online |
| listWebServiceClientPolicies | List web service client port policies information for an application or SOA composite. | Online |
| listWebServicePolicies | List web service port policy information for a web service in an application or SOA composite. | Online |
| setWSMPolicyOverride | Configure override properties for a policy that is attached to the selected policy subject or policy set document within a session. | Online |

Use the WLST commands listed in Table 3–7 to manage Java EE Web Services (or clients) directly attached policies.

> **Note:** The commands listed in Table 3–7 have an `application` argument.
>
> In an multi-tenant environment, if you intend to target a specific application instance within a tenant's partition, then you must include the partition name as part of the application as follows:
>
> `/domain/server/application#version$partition`
>
> However, if you are targeting a domain-scoped application, then you do not have to include the partition name. You can use the `application` argument as follows:
>
> `/domain/server/application#version`

***Table 3–7    Java EE Web Services (or Clients) - WLST Commands for Direct Policy Attachments***

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| attachWebServiceClientPolicies | Attach multiple policies to a web service client port of an application or SOA composite. | Online |
| attachWebServiceClientPolicy | Attach an OWSM policy to a web service client port of an application or SOA composite. | Online |
| attachWebServicePolicies | Attach multiple policies to a web service port of an application or SOA composite. | Online |
| attachWebServicePolicy | Attach a policy to a web service port of an application or SOA composite. | Online |
| detachWebServiceClientPolicies | Detach multiple policies from a web service client port of an application or SOA composite. | Online |
| detachWebServiceClientPolicy | Detach a policy from a web service client port of an application or SOA composite. | Online |
| detachWebServicePolicies | Detach multiple OWSM policies from a web service port of an application or SOA composite | Online |
| detachWebServicePolicy | Detach an OWSM policy from a web service port of an application or SOA composite. | Online |
| enableWebServiceClientPolicies | Enable or disable multiple policies of a web service client port of an application or SOA composite. | Online |
| enableWebServiceClientPolicy | Enable or disable a policy of a web service client port of an application or SOA composite. | Online |
| enableWebServicePolicies | Enable or disable multiple policies attached to a port of a web service application or SOA composite. | Online |
| enableWebServicePolicy | Enable or disable a policy attached to a port of a web service application or SOA composite. | Online |

*Table 3–7   (Cont.)  Java EE Web Services (or Clients) - WLST Commands for Direct Policy Attachments*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| listAvailableWebServicePolicies | Display a list of all the available OWSM policies by category or subject type. | Online |
| listWebServiceClientPolicies | List web service client port policies information for an application or SOA composite. | Online |
| listWebServicePolicies | List web service port policy information for a web service in an application or SOA composite. | Online |

## attachWebServiceClientPolicies

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure web services.
>
> For Oracle Infrastructure Web Services, this command has been deprecated. It is recommended that you use the `attachWSMPolicies` command, as described in "attachWSMPolicies" on page 3-64. The following examples show how to migrate to use the `attachWSMPolicies` command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>attachWebServiceClientPolicies
> ('/base_domain/server1/jwsclient_
> 1#1.1.0','WssUsernameClient','wsconn',
> 'WssUsernameClient','JRFWssUsernamePort',["oracle/wss_username_
> token_client_policy","oracle/log_policy"])
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig>attachWSMPolicies(["oracle/wss_
> username_token_client_policy","oracle/log_policy"])
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Attaches multiple policies to a web service client port of an application or SOA composite.

The `policyURIs` are validated through the OWSM Policy Manager APIs if the `wsm-pm` application is installed on WebLogic Server and is available.

For Java EE (`wls`) module types only: If the policies that you specify in this command are already attached or exist, then this command enables the policies that are already attached (if they are disabled), and attaches the others.

If the `wsm-pm` application is not installed or is not available, this command is not executed.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application.

### Syntax

```
attachWebServiceClientPolicies(application,moduleOrCompName,moduleType,
serviceRefName,portInfoName,policyURIs,[subjectType=None] )
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to attach OWSM client policies to the web service client port. For example, `/domain/server/application#version_number` |
| | To attach policies to a client port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, `HelloWorld[1.0]`) for which you want to attach the policies to the client port. |
| | To attach policies to a client port of a SOA composite, the composite name is required (for example, `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`. |
| *moduleType* | Module type. Valid options are: |
| | ■    `soa`—SOA composite. |
| | ■    `web`—Oracle Infrastructure web services packaged as a Web module (including an EJB). |
| | ■    `wls`—Java EE web services. |
| | ■    `wsconn`—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy, or WebCenter client. |
| | **Note**: The `web` and `wsconn` module types are deprecated for this release. |
| *serviceRefName* | The service reference name of the application or composite. |
| *portInfoName* | The client port to which you want to attach the OWSM client policy. |
| *policyURI* | The OWSM policy name URIs, for example, `["oracle/log_policy","oracle/wss_username_token_client_policy"]` |
| | If the policies that you specify in this command are already attached or exist, then this command enables the policies that are already attached (if they are disabled), and attaches the others. |
| *subjectType* | Optional. Policy subject type. Valid options are: |
| | ■    `P`—Port. The default is `P`. |
| | ■    `O`—Not supported in this release. |

## Examples

The following example attaches the policy `oracle/log_policy` to the client port `HelloWorld_pt` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>attachWebServiceClientPolicies
(None, 'default/HelloWorld[1.0]','soa','client','HelloWorld_pt',["oracle/wss_
username_token_client_policy","oracle/log_policy"])
```

The following example attaches the policies `oracle/wss10_saml20_token_client_policy` and `oracle/wss11_message_protection_client_policy` to the client port `UpperCaseImplPort` in the Java EE Web module `owsm_mbean.resouce_pattern.web.ClientJWS/sei2`.

```
wls:/wls-domain/serverConfig>attachWebServiceClientPolicies
('/wls-domain/AdminServer/ClientJWS','owsm_mbean.resouce_
pattern.web.ClientJWS/sei2','wls','owsm_mbean.resouce_
pattern.web.ClientJWS/sei2','UpperCaseImplPort',["oracle/wss10_saml20_token_
client_policy","oracle/wss11_message_protection_client_policy"])
```

## attachWebServiceClientPolicy

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, it is recommended that you use the `attachWSMPolicy` command, as described in "attachWSMPolicy" on page 3-65. The following examples show how to migrate to use the `attachWSMPolicy` command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>attachWebServiceClientPolicy
> ('/base_domain/server1/jwsclient_
> 1#1.1.0','WssUsernameClient','wsconn',
> 'WssUsernameClient','JRFWssUsernamePort',"oracle/wss_username_
> token_client_policy")
> ```
>
> 12*c*:
>
> ```
> wls:/wls-domain/serverConfig>attachWSMPolicy("oracle/wss_username_
> token_client_policy")
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Attaches an OWSM policy to a web service client port of an application or SOA composite.

The policyURI is validated through the OWSM Policy Manager APIs if the `wsm-pm` application is installed on WebLogic Server and is available.

For Java EE (`wls`) module types only: If the PolicyURI that you specify in this command already is attached or exists, then this command enables the policy if it is disabled.

If the `wsm-pm` application is not installed or is not available, this command is not executed.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application.

### Syntax

```
attachWebServiceClientPolicy(application,moduleOrCompName,moduleType,
serviceRefName, portInfoName, policyURI, [subjectType=None] )
```

| Argument | Definition |
|----------|------------|
| *application* | Name and path of the application for which you want to attach a policy to the web service client port. For example, /domain/server/application#version_number. |
| | To attach a policy to a client port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to attach the policy to the client port. |
| | To attach a policy to a client port of a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |
| *moduleType* | Module type. Valid options are: |
| | ■ soa—SOA composite. |
| | ■ web—Oracle Infrastructure web services packaged as a Web module (including an EJB). |
| | ■ wls—Java EE web services. |
| | ■ wsconn—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy. |
| | **Note**: The web and wsconn module types are deprecated for this release. |
| *serviceRefName* | The service reference name of the application or composite. |
| *portInfoName* | The client port to which you want to attach the OWSM client policy. |
| *policyURI* | The OWSM policy name URI, for example, oracle/wss_username_token_client_policy" |
| | If the policy that you specify is already attached or exists, then this command enables the policy if it is disabled. |
| *subjectType* | Optional. Policy subject type. Valid options are: |
| | ■ P—Port. The default is P. |
| | ■ O—Not supported in this release. |

## Examples

The following example attaches the client policy oracle/log_policy to the client port HelloWorld_pt in the SOA composite default/HelloWorld[1.0].

```
wls:/wls-domain/serverConfig>attachWebServiceClientPolicy
(None, 'default/HelloWorld[1.0]','soa','client','HelloWorld_pt','oracle/log_
policy')
```
The following example attaches the oracle/wss_username_token_client_policy client policy to the Java EE web service client port UpperCaseImplPort of the Web module owsm_mbean.resouce_pattern.web.ClientJWS/sei2. The web service is part of the application ClientJWS.

```
wls:/wls-domain/serverConfig> attachWebServiceClientPolicy
('/wls-domain/AdminServer/ClientJWS','owsm_mbean.resouce_
pattern.web.ClientJWS/sei2', 'wls','owsm_mbean.resouce_
pattern.web.ClientJWS/sei2', 'UpperCaseImplPort', "oracle/wss_username_token_
client_policy")
```

## attachWebServicePolicies

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, it is recommended that you use the attachWSMPolicies command, as described in "attachWSMPolicies" on page 3-64. The following examples show how to migrate to use the attachWSMPolicies command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig> attachWebServicePolicies
> ('/base_domain/server1/HelloWorld#1_0','j2wbasicPolicy','web',
> '{http://namespace/}WssUsernameService','JRFWssUsernamePort',
> ["oracle/log_policy", "oracle/wss_username_token_service_policy"])
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig> attachWSMPolicies["oracle/log_
> policy", "oracle/wss_username_token_service_policy"])
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Attaches multiple policies to a web service port of an application or SOA composite.

The policyURIs are validated through the OWSM Policy Manager APIs if the wsm-pm application is installed on WebLogic Server and is available.

For Java EE (wls) module types only: if any of the policies that you specify in this command are already attached or exist, then this command enables the policies that are already attached (if they are disabled), and attaches the others.

If the wsm-pm application is not installed or is not available, this command is not executed.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application.

### Syntax

attachWebServicePolicies(application, moduleOrCompName, moduleType, serviceName, subjectName,policyURIs,[subjectType=None])

| Argument | Definition |
|---|---|
| *application* | Name and path of the application to which you want to attach the web service policies. For example, /domain/server/application#version_number |
| | To attach the policies to a port of a web service application, this argument is required. |

| Argument | Definition |
|---|---|
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) to which you want to attach web service policies. |
| | To attach the policies to a port of a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |
| *moduleType* | Module type. Valid options are:<br><br>■ soa—SOA composite.<br><br>■ web—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br><br>■ wls—Java EE web services.<br><br>**Note**: The web module type is deprecated for this release. |
| *serviceName* | Name of the web service in the application or SOA composite. For example, {http://namespace/}serviceName. Note that the namespace ({http://namespace/}) should not be included for a SOA composite. |
| *subjectName* | Name of the policy subject, port, or operation. |
| *policyURIs* | List of OWSM policy name URIs, for example, ["oracle/log_policy","oracle/wss_username_token_service_policy"] |
| | If any of the policies that you specify are already attached or exist, then this command enables the policies that are already attached (if they are disabled), and attaches the others. |
| *subjectType* | Optional. Policy subject type. Valid options are:<br><br>■ P—Port. The default is P.<br><br>■ O—Not supported in this release. |

## Example

The following example attaches the policies 'oracle/binding_authorization_denyall_policy', 'oracle/wss_username_token_service_policy' to the port helloWorldJaxwsSoapHttpPort of the Web module helloWorldJaxws. The Java EE web service is part of the application helloWorldJaxws for the server AdminServer in the domain wls-domain.

```
 wls:wls-domain/ServerConfig>attachWebServicePolicies
('/wls-domain/AdminServer/helloWorldJaxws','helloWorldJaxws#1!helloWorldJaxws',
'wls','helloWorldJaxws', 'helloWorldJaxwsSoapHttpPort, ['oracle/binding_
authorization_denyall_policy', 'oracle/wss_username_token_service_policy'])
```

## attachWebServicePolicy

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, it is recommended that you use the `attachWSMPolicy` command, as described in "attachWSMPolicy" on page 3-65. The following examples show how to migrate to use the `attachWSMPolicy` command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig> attachWebServicePolicy
> ('/base_domain/server1/HelloWorld#1_0','j2wbasicPolicy','web',
> '{http://namespace/}WssUsernameService','JRFWssUsernamePort','oracl
> e/wss_username_token_service_policy')
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig> attachWSMPolicy('oracle/wss_username_
> token_service_policy')
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Attaches a policy to a web service port of an application or SOA composite.

The policyURI is validated through the OWSM Policy Manager APIs if the `wsm-pm` application is installed on WebLogic Server and is available.

For Java EE (`wls`) module types only: If the PolicyURI that you specify in this command already is attached or exists, then this command enables the policy if it is disabled.

If the `wsm-pm` application is not installed or is not available, this command is not executed.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application.

### Syntax

```
attachWebServicePolicy(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURI, [subjectType=None])
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application to which you want to attach a web service policy. For example, /domain/server/application#version_number |
| | To attach a policy to a port of a web service application, this argument is required. |

| Argument | Definition |
|---|---|
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, `HelloWorld[1.0]`) to which you want to attach a web service policy. |
| | To attach a policy to a port of a SOA composite, the composite name is required (for example, `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`. |
| *moduleType* | Module type. Valid options are: |
| | ■ `soa`—SOA composite. |
| | ■ `web`—Oracle Infrastructure web services packaged as a Web module (including an EJB). |
| | ■ `wls`—Java EE web services. |
| | **Note**: The `web` module type is deprecated for this release. |
| *serviceName* | Name of the web service in the application or SOA composite. For example, `{http://namespace/}serviceName`. Note that the namespace (`{http://namespace/}`) should not be included for a SOA composite. |
| *subjectName* | Name of the policy subject, port, or operation. |
| *policyURI* | OWSM policy name URI, for example, `'oracle/log_policy'` |
| *subjectType* | Optional. Policy subject type. Valid options are: |
| | ■ `P`—Port. The default is `P`. |
| | ■ `O`—Not supported in this release. |

## Examples

The following example attaches the policy `oracle/log_policy` to the port `HelloWorld_pt` of the service `HelloService` in the SOA composite `default/HelloWorld[1.0]`. Note that the namespace (`{http://namespace/}`) should not be included for a SOA composite.

```
wls:/wls-domain/serverConfig>attachWebServicePolicy(None,
'default/HelloWorld[1.0]','soa','HelloService','HelloWorld_pt','oracle/log_
policy')
```

The following example attaches the policy `oracle/wss_username_token_service_policy` to the port `helloWorldJaxwsSoapHttpPort` of the Java EE web service `helloWorldJaxws`.

```
wls:wls-domain/serverConfig> attachWebServicePolicy
('/wls-domain/AdminServer/helloWorldJaxws','helloWorldJaxws#1!helloWorldJaxws',
'wls','helloWorldJaxws', 'helloWorldJaxwsSoapHttpPort', 'oracle/wss_username_
token_service_policy')
```

A web service cannot contain both a WebLogic web service policy and an Oracle web service policy. If you have a web service with a WebLogic web service policy, you must first detach it before attaching the Oracle web service policy. The following example detaches the WebLogic web service policy `Wssp1.2-2007-Saml2.0-SenderVouches-Wss1.1.xml` from the port `SimplePort` in the Java EE web service `SimpleEjbService` and then attaches the Oracle web service policy `oracle/wss_username_token_service_policy`.

```
wls:wls-domain/serverConfig>detachWebServicePolicy('/wls-domain/AdminServer/Simple
JAXWS','SimpleJAXWS#1!SimpleEjbService', 'wls','SimpleEjbService',
'SimplePort','policy:Wssp1.2-2007-Saml2.0-SenderVouches-Wss1.1.xml')
```

```
wls:wls-domain/serverConfig>attachWebServicePolicy('/wls-domain/AdminServer/Simple
JAXWS','SimpleJAXWS#1!SimpleEjbService', 'wls','SimpleEjbService', 'SimplePort',
'oracle/wss_username_token_service_policy')
```

> **Note:** The `detachWebServicePolicy` WLST command allows you to detach WebLogic web service policies from a web service. However, you cannot use the `attachWebServicePolicy` WLST command to attach WebLogic web service policies. To attach WebLogic web service policies to a web service, you must use the WebLogic Administration Console.

## attachWSMPolicies

> **Note:** This command applies to Oracle Infrastructure and RESTful web services. It does not apply to Java EE web services in this release.

Command Category: Policy Management

Use with WLST: Online

### Description

Within a session, attaches multiple policies, identified by specified the URIs, to the selected policy subject.

You must start a session and select the policy set (`selectWSMPolicySet`) or policy subject (`selectWSMPolicySubject`) before initiating the command. However, if `attachWSMPolicies` is issued when creating or cloning a policy set, there is no need to select the policy set because it is already selected. If there is no current session and no policy subject selected, an error is displayed.

### Syntax

```
attachWSMPolicies(uris)
```

| Element | Description |
|---------|-------------|
| *uris* | List of OWSM policy name URIs, for example, `["oracle/log_policy","oracle/wss_username_token_service_policy"]` |

### Examples

The following example attaches the policies `oracle/log_policy` and `oracle/wss_username_token_service_policy`. It assumes that you have already selected a policy subject.

```
wls:/wls-domain/serverConfig>attachWSMPolicies(["oracle/log_policy", "oracle/wss_username_token_service_policy"])
```

# attachWSMPolicy

> **Note:** This command applies to Oracle Infrastructure and RESTful web services. It does not apply to Java EE web services in this release.

Command Category: Policy Management

Use with WLST: Online

## Description

Within a session, attaches a policy, identified by the specified URI, to the selected policy subject or policy set.

You must start a session and select the policy set (`selectWSMPolicySet`) or policy subject (`selectWSMPolicySubject`) before initiating the command. However, if `attachWSMPolicy` is issued when creating or cloning a policy set, there is no need to select the policy set because it is already selected. If there is no current session and no policy subject is selected, an error is displayed.

## Syntax

```
attachWSMPolicy(uri)
```

| Argument | Definition |
|---|---|
| *uri* | OWSM policy name URI, for example, `'oracle/log_policy'` |

## Examples

The following example attaches the policy `oracle/wss_username_token_service_policy`. It assumes that you have already selected a web service port, a web service client port, or a current policy set.

```
wls:/wls-domain/serverConfig>attachWSMPolicy('oracle/wss_username_token_service_policy')
```

## detachWebServiceClientPolicies

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, it is recommended that you use the detachWSMPolicies command, as described in "detachWSMPolicies" on page 3-74. The following examples show how to migrate to use the detachWSMPolicies command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>detachWebServiceClientPolicies
> ('/base_domain/server1/jwsclient_
> 1#1.1.0','WssUsernameClient','wsconn',
> 'WssUsernameClient','JRFWssUsernamePort',
> ["oracle/log_policy","oracle/wss_username_token_client_policy"])
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig>detachWSMPolicies(["oracle/log_
> policy","oracle/wss_username_token_client_policy"])
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Detaches multiple policies from a web service client port of an application or SOA composite.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application.

### Syntax

```
detachWebServiceClientPolicies(application,moduleOrCompName,moduleType,
serviceRefName,portInfoName,policyURIs,[subjectType=None] )
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to detach multiple policies from a web service client port. For example, /domain/server/application#version_number |
| | To detach multiple policies from a client port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to detach multiple policies from a client port. |
| | To detach multiple policies from a client port for a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |

| Argument | Definition |
|---|---|
| *moduleType* | Module type. Valid options are:<br><br>■ soa—SOA composite.<br><br>■ web—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br><br>■ wls—Java EE web services.<br><br>■ wsconn—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy.<br><br>**Note**: The web and wsconn module types are deprecated for this release. |
| *serviceRefName* | The service reference name of the application or composite. |
| *portInfoName* | The client port from which you want to detach the OWSM client policy. |
| *policyURI* | The OWSM policy name URI, for example, oracle/wss_username_token_client_policy"<br><br>If the policy specified is not attached, an error message is displayed and/or an exception is thrown. |
| *subjectType* | Optional. Policy subject type. Valid options are:<br><br>■ P—Port. The default is P.<br><br>■ O—Not supported in this release. |

## Example

The following example detaches the client policies oracle/wss10_saml20_token_client_policy and oracle/wss11_message_protection_client_policy of the port UpperCaseImplPort of the Java EE web service module owsm_mbean.resouce_pattern.web.ClientJWS/sei2.

```
wls:/wls-domain/serverConfig>detachWebServiceClientPolicies('/wls-domain/AdminServ
er/ClientJWS','owsm_mbean.resouce_pattern.web.ClientJWS/sei2','wls','owsm_
mbean.resouce_pattern.web.ClientJWS/sei2','UpperCaseImplPort',["oracle/wss10_
saml20_token_client_policy","oracle/wss11_message_protection_client_policy"])
```

## detachWebServiceClientPolicy

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, it is recommended that you use the detachWSMPolicy command, as described in "detachWSMPolicy" on page 3-75. The following examples show how to migrate to use the detachWSMPolicy command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>detachWebServiceClientPolicy
> ('/base_domain/server1/jwsclient_
> 1#1.1.0','WssUsernameClient','wsconn',
> 'WssUsernameClient','JRFWssUsernamePort','oracle/wss_username_
> token_client_policy')
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig>detachWSMPolicy('oracle/wss_username_
> token_client_policy')
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Detaches a policy from a web service client port of an application or SOA composite.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application.

### Syntax

```
detachWebServiceClientPolicy(application,moduleOrCompName,moduleType,
serviceRefName, portInfoName, policyURI, [subjectType=None] )
```

| Argument | Definition |
|----------|------------|
| *application* | Name and path of the application for which you want to detach a policy from a web service client port. For example, /domain/server/application#version_number |
| | To detach a policy from a client port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to detach the policy from a client port. |
| | To detach a policy from a client port of a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |

| Argument | Definition |
|----------|------------|
| *moduleType* | Module type. Valid options are: <br><br>■ soa—SOA composite. <br><br>■ web—Oracle Infrastructure web services packaged as a Web module (including an EJB). <br><br>■ wls—Java EE web services. <br><br>■ wsconn—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy. <br><br>**Note**: The web and wsconn module types are deprecated for this release. |
| *serviceRefName* | The service reference name of the application or composite. |
| *portInfoName* | The client port from which you want to detach the OWSM client policy. |
| *policyURI* | The OWSM policy name URI, for example, oracle/wss_username_token_client_policy" <br><br>If the policy specified is not attached, an error message is displayed and/or an exception is thrown. |
| *subjectType* | Optional. Policy subject type. Valid options are: <br><br>■ P—Port. The default is P. <br><br>■ O—Not supported in this release. |

### Examples

The following example detaches the client policy oracle/log_policy from the client port HelloWorld_pt in the SOA composite default/HelloWorld[1.0].

```
wls:/wls-domain/serverConfig>detachWebServiceClientPolicy(None,
'default/HelloWorld[1.0]','soa','client','HelloWorld_pt','oracle/log_policy' )
```
The following command detaches the client policy oracle/wss_username_token_client_policy from the client port UpperCaseImplPort in the Java EE client module wsm_mbean.resouce_pattern.web.ClientJWS/sei2.

```
wls:/wls-domain/serverConfig>
detachWebServiceClientPolicy('/wls-domain/AdminServer/ClientJWS', 'owsm_
mbean.resouce_pattern.web.ClientJWS/sei2', 'wls', 'owsm_mbean.resouce_
pattern.web.ClientJWS/sei2', 'UpperCaseImplPort', "oracle/wss_username_token_
client_policy")
```

## detachWebServicePolicies

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, it is recommended that you use the `detachWSMPolicies` command, as described in "detachWSMPolicies" on page 3-74. The following examples show how to migrate to use the `detachWSMPolicies` command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>detachWebServicePolicies
> ('/base_domain/server1/HelloWorld#1_0','j2wbasicPolicy','web',
> '{http://namespace/}WssUsernameService','JRFWssUsernamePort',
> ["oracle/log_policy","oracle/wss_username_token_service_policy"])
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig>detachWSMPolicies(["oracle/log_
> policy","oracle/wss_username_token_service_policy"])
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Detaches multiple OWSM policies from a web service port of an application or SOA composite.

If the `wsm-pm` application is not installed or is not available, this command is not executed.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application.

### Syntax

```
detachWebServicePolicies(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURIs,[subjectType=None])
```

| Argument | Definition |
|----------|------------|
| *application* | Name and path of the application from which you want to detach the web service policies. For example, /domain/server/application#version_number |
| | To detach policies from a port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) from which you want to detach the web service policies. |
| | To detach policies from a port of a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |

| Argument | Definition |
|---|---|
| *moduleType* | Module type. Valid options are: <br><br>■   soa—SOA composite. <br><br>■   web—Oracle Infrastructure web services packaged as a Web module (including an EJB). <br><br>■   wls—Java EE web services. <br><br>**Note**: The web module type is deprecated for this release. |
| *serviceName* | Name of the web service in the application or SOA composite. For example, {http://namespace/}serviceName. Note that the namespace ({http://namespace/}) should not be included for a SOA composite. |
| *subjectName* | Name of the policy subject, port, or operation. |
| *policyURIs* | List of OWSM policy name URIs, for example, ["oracle/log_policy","oracle/wss_username_token_service_policy"] <br><br>If a policyURI specified is not attached, an error message is displayed and/or an exception is thrown. |
| *subjectType* | Optional. Policy subject type. Valid options are: <br><br>■   P—Port. The default is P. <br><br>■   O—Not supported in this release. |

### Example

The following example detaches the policies "oracle/binding_authorization_denyall_policy", "oracle/wss_username_token_service_policy" from the port helloWorldJaxwsSoapHttpPort of the Java EE Web module helloWorldJaxws. The web service is part of the application helloWorldJaxws for the server AdminServer in the domain wls-domain.

```
wls:/wls-domain/serverConfig>detachWebServicePolicies
('/wls-domain/AdminServer/helloWorldJaxws','helloWorldJaxws#1!helloWorldJaxws',
'wls','helloWorldJaxws', 'helloWorldJaxwsSoapHttpPort', ["oracle/binding_
authorization_denyall_policy", "oracle/wss_username_token_service_policy"])
```

## detachWebServicePolicy

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, it is recommended that you use the detachWSMPolicy command, as described in "detachWSMPolicy" on page 3-75. The following examples show how to migrate to use the detachWSMPolicy command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>detachWebServicePolicy('/base_
> domain/server1/HelloWorld#1_
> 0','j2wbasicPolicy','web','{http://namespace/}WssUsernameService','
> JRFWssUsernamePort','oracle/wss_username_token_service_policy')
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig>detachWSMPolicy('oracle/wss_username_
> token_service_policy')
> ```

Command Category: Policy Management

Use with WLST: Online

## Description

Detaches an OWSM policy from a web service port of an application or SOA composite.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application.

## Syntax

```
detachWebServicePolicy(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURI, [subjectType=None])
```

| Argument | Definition |
| --- | --- |
| *application* | Name and path of the application from which you want to detach a web service policy. For example, /domain/server/application#version_number |
| | To detach a policy from a port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) from which you want to detach a web service policy. |
| | To detach a policy from a port of a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |

| Argument | Definition |
|---|---|
| *moduleType* | Module type. Valid options are:<br><br>■ soa—SOA composite.<br><br>■ web—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br><br>■ wls—Java EE web services.<br><br>**Note**: The web module type is deprecated for this release. |
| *serviceName* | Name of the web service in the application or SOA composite. For example, {http://namespace/}serviceName. Note that the namespace ({http://namespace/}) should not be included for a SOA composite. |
| *subjectName* | Name of the policy subject, port, or operation. |
| *policyURI* | OWSM policy name URI, for example, 'oracle/log_policy'<br><br>If the policy specified is not attached, an error message is displayed and/or an exception is thrown. |
| *subjectType* | Optional. Policy subject type. Valid options are:<br><br>■ P—Port. The default is P.<br><br>■ O—Not supported in this release. |

### Examples

The following example detaches the policy oracle/log_policy from the port HelloWorld_pt of the service HelloService in the SOA composite default/HelloWorld[1.0]. Note that the namespace ({http://namespace/}) should not be included for a SOA composite.

```
wls:/wls-domain/serverConfig>detachWebServicePolicy(None,
'default/HelloWorld[1.0]', 'soa','HelloService','HelloWorld_pt','oracle/log_
policy')
```

The following example detaches the policy oracle/wss_username_token_service_policy from the port helloWorldJaxwsSoapHttpPort of the service helloWorldJaxws in the Java EE web service wls-domain/AdminServer/helloWorldJaxws.

```
wls:/wls-domain/serverConfig>detachWebServicePolicy
('/wls-domain/AdminServer/helloWorldJaxws','helloWorldJaxws#1!helloWorldJaxws',
'wls','helloWorldJaxws', 'helloWorldJaxwsSoapHttpPort', 'oracle/wss_username_
token_service_policy')
```

## detachWSMPolicies

> **Note:** This command applies to Oracle Infrastructure and RESTful web services. It does not apply to Java EE web services in this release.

Command Category: Policy Management

Use with WLST: Online

### Description

Within a session, detaches multiple policies, identified by an array of URIs or index values, from the selected policy subject.

You must start a session and select the policy set (selectWSMPolicySet) or policy subject (selectWSMPolicySubject) before initiating the command. If there is no current session and no policy subject selected, an error is displayed.

### Syntax

    detachWSMPolicies(uris)

| Argument | Definition |
| --- | --- |
| uris | Array of URIs or index values specifying the policies to detach from a policy subject. For example, ["oracle/log_policy","oracle/wss_username_token_service_policy"] |
| | If the specified policy URIs are not attached, an error message is displayed and/or an exception is thrown. |

### Examples

The following example detaches the OWSM logging policy and username token service policy from the current policy subject:

    wls:/wls-domain/serverConfig>**detachWSMPolicies(["oracle/log_policy","oracle/wss_username_token_service_policy"])**

The following example uses the index values of the OWSM logging policy and username token service URIs to detach them from the current policy subject

    wls:/wls-domain/serverConfig>**detachWSMPolicies('1','3')**

## detachWSMPolicy

> **Note:** This command applies to Oracle Infrastructure and RESTful web services. It does not apply to Java EE web services in this release.

Command Category: Policy Management

Use with WLST: Online

### Description

Within a session, detaches a policy, identified by the specified URI or index value, from the selected policy subject.

You must start a session and select the policy set (`selectWSMPolicySet`) or policy subject (`selectWSMPolicySubject`) before initiating the command. If there is no current session and no policy subject selected, an error is displayed

Issuing this command outside of a session containing a policy subject that is being created or modified will result in an error.

### Syntax

```
detachWSMPolicy(uri)
```

| Argument | Definition |
|----------|------------|
| *uri* | URI or index value specifying the policy to detach from a policy subject. For example, `'oracle/log_policy'`. |
| | If the specified policy URI is not attached, an error message is displayed and/or an exception is thrown. |

### Examples

The following example detaches the OWSM logging policy from the current policy subject.

```
wls:/wls-domain/serverConfig>detachWSMPolicy('oracle/log_policy')
```

The following example uses the index value of the OWSM logging policy's URI to detach it from the current policy subject.

```
wls:/wls-domain/serverConfig>detachWSMPolicy('1')
```

## enableWebServiceClientPolicies

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, it is recommended that you use the `enableWSMPolicies` command, as described in "enableWSMPolicies" on page 3-84. The following examples show how to migrate to use the `enableWSMPolicies` command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>enableWebServiceClientPolicies
> ('/base_domain/server1/jwsclient_
> 1#1.1.0','WssUsernameClient','wsconn',
> 'WssUsernameClient','JRFWssUsernamePort',
> ["oracle/log_policy", "oracle/wss_username_token_client_policy"],
> true )
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig>enableWSMPolicies(["oracle/log_
> policy", "oracle/wss_username_token_client_policy"], true )
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Enables or disables multiple policies of a web service client port of an application or SOA composite.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application

### Syntax

```
enableWebServiceClientPolicies(application,moduleOrCompName,moduleType,
serviceRefName,portInfoName,policyURIs,[enable],[subjectType=None] )
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to enable or disable multiple policies of a web service client port. For example, /domain/server/application#version_number |
| | To enable or disable multiple policies of a client port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to enable or disable multiple policies of a client port. |
| | To enable or disable multiple policies of a client port for a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |

| Argument | Definition |
|---|---|
| *moduleType* | Module type. Valid options are:<br><br>■  soa—SOA composite.<br><br>■  web—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br><br>■  wls—Java EE web services.<br><br>■  wsconn—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy.<br><br>**Note**: The web and wsconn module types are deprecated for this release. |
| *serviceRefName* | The service reference name of the application or composite. |
| *portInfoName* | The name of the client port to which you want to attach the OWSM client policies. |
| *policyURIs* | The list of OWSM policy name URIs, for example, ["oracle/log_policy","oracle/wss_username_token_client_policy"]. |
| *enable* | Optional. Specifies whether to enable or disable the policies. Valid options are:<br><br>■  true—Enables the policy. The default is true.<br><br>■  false—Disables the policy.<br><br>If you omit this argument, the policies are enabled. |
| *subjectType* | Optional. Policy subject type. Valid options are:<br><br>■  P—Port. The default is P.<br><br>■  O—Not supported in this release. |

### Example

The following example enables the client policies oracle/wss10_saml20_token_client_policy and oracle/wss11_message_protection_client_policy of the port UpperCaseImplPort of the Java EE web service module owsm_mbean.resouce_pattern.web.ClientJWS/sei2.

```
wls:/wls-domain/serverConfig>enableWebServiceClientPolicies('/wls-domain/AdminServer/ClientJWS','owsm_mbean.resouce_pattern.web.ClientJWS/sei2','wls','owsm_mbean.resouce_pattern.web.ClientJWS/sei2','UpperCaseImplPort',["oracle/wss10_saml20_token_client_policy","oracle/wss11_message_protection_client_policy"],true)
```

## enableWebServiceClientPolicy

> **Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, it is recommended that you use the enableWSMPolicy command, as described in "enableWSMPolicy" on page 3-85. The following examples show how to migrate to use the enableWSMPolicy command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>enableWebServiceClientPolicy
> ('/base_domain/server1/jwsclient_
> 1#1.1.0','WssUsernameClient','wsconn',
> 'WssUsernameClient','JRFWssUsernamePort', "oracle/wss_username_
> token_client_policy",true)
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig>enableWSMPolicy("oracle/wss_username_
> token_client_policy",true)
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Enables or disables a policy of a web service client port of an application or SOA composite.

> **Note:** Policy changes made using this WLST command are only effective after you restart your application.

### Syntax

```
enableWebServiceClientPolicy(application,moduleOrCompName,moduleType,
serviceRefName,portInfoName,policyURI,[enable],[subjectType=None] )
```

| Argument | Definition |
|----------|------------|
| *application* | Name and path of the application for which you want to enable or disable a policy of a web service client port. For example, /domain/server/application#version_number |
| | To enable or disable a policy of a client port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to enable or disable a policy of a client port. |
| | To enable or disable a policy of a client port for a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |

| Argument | Definition |
|---|---|
| *moduleType* | Module type. Valid options are:<br><br>■ soa—SOA composite.<br><br>■ web—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br><br>■ wls—Java EE web services.<br><br>■ wsconn—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy.<br><br>**Note**: The web and wsconn module types are deprecated for this release. |
| *serviceRefName* | The service reference name of the application or composite. |
| *portInfoName* | The name of the client port to which you want to attach the OWSM client policy. |
| *policyURI* | The OWSM policy name URI, for example, oracle/wss_username_token_client_policy" |
| *enable* | Optional. Specifies whether to enable or disable the policy. Valid options are:<br><br>■ true—Enables the policy. The default is true.<br><br>■ false—Disables the policy.<br><br>If you omit this argument, the policy is enabled. |
| *subjectType* | Optional. Policy subject type. Valid options are:<br><br>■ P—Port. The default is P.<br><br>■ O—Not supported in this release. |

### Examples

The following example enables the client policy oracle/log_policy of the client port HelloWorld_pt in the SOA composite default/HelloWorld[1.0].

```
wls:/wls-domain/serverConfig>enableWebServiceClientPolicy(None,
'default/HelloWorld[1.0]','soa','client','HelloWorld_pt','oracle/log_policy')
```

The following example disables the client policy oracle/log_policy of the client port HelloWorld_pt in the SOA composite default/HelloWorld[1.0].

```
wls:/wls-domain/serverConfig>enableWebServiceClientPolicy(None,
'default/HelloWorld[1.0]','soa','client','HelloWorld_pt','oracle/log_policy',
false )
```

The following example disables the client policy oracle/wss_username_token_client_policy on the client port UpperCaseImplPort in the Java EE Web module owsm_mbean.resouce_pattern.web.ClientJWS/sei2.

```
wls:/wls-domain/serverConfig>enableWebServiceClientPolicy('/wls-domain/AdminServer
/ClientJWS', 'owsm_mbean.resouce_pattern.web.ClientJWS/sei2', 'wls', 'owsm_
mbean.resouce_pattern.web.ClientJWS/sei2', 'UpperCaseImplPort', "oracle/wss_
username_token_client_policy", false)
```

## enableWebServicePolicies

---

**Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.

For Oracle Infrastructure Web Services, it is recommended that you use the `enableWSMPolicies` command, as described in "enableWSMPolicies" on page 3-84. The following examples show how to migrate to use the `enableWSMPolicies` command.

11*g* Release:

```
wls:/wls-domain/serverConfig> enableWebServicePolicies
('/base_domain/server1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort',["orac
le/log_policy", "oracle/wss_username_token_service_policy"],true)
```

12*c* Release:

```
wls:/wls-domain/serverConfig> enableWSMPolicies(["oracle/log_
policy","oracle/wss_username_token_service_policy"],true)
```

---

Command Category: Policy Management

Use with WLST: Online

## Description

Enables or disables multiple policies attached to a port of a web service application or SOA composite.

If the `policyURIs` that you specify in this command are not attached to the port, an error message is displayed and/or an exception is thrown.

---

**Note:** Policy changes made using this WLST command are only effective after you restart your application.

---

## Syntax

```
enableWebServicePolicies(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURIs,[enable],[subjectType=None] ))
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to enable the web service policies. For example, /domain/server/application#version_number |
| | To enable policies that are attached to a port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to enable web service policies. |
| | To enable policies that are attached to a port of a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |

| Argument | Definition |
|----------|-----------|
| *moduleType* | Module type. Valid options are: <br>■ `soa`—SOA composite. <br>■ `web`—Oracle Infrastructure web services packaged as a Web module (including an EJB). <br>■ `wls`—Java EE web services. <br> **Note**: The `web` module type is deprecated for this release. |
| *serviceName* | Name of the web service in the application or SOA composite.For example, `{http://namespace/}serviceName`. Note that the namespace (`{http://namespace/}`) should not be included for a SOA composite. |
| *subjectName* | Name of the policy subject, port, or operation. |
| *policyURIs* | List of OWSM policy name URIs, for example, `["oracle/log_policy","oracle/wss_username_token_service_policy"]` <br> If the `policyURIs` that you specify are not attached, an error message is displayed and/or an exception is thrown. |
| *enable* | Optional. Specifies whether to enable or disable the policies. Valid options are: <br>■ `true`—Enables the policies. The default is `true`. <br>■ `false`—Disables the policies. <br> If you omit this argument, the policies are enabled. |
| *subjectType* | Optional. Policy subject type. Valid options are: <br>■ `P`—Port. The default is `P`. <br>■ `O`—Not supported in this release. |

## Example

The following example disables the policies `["oracle/binding_authorization_denyall_policy","oracle/wss_username_token_service_policy"]` attached to the port `helloWorldJaxwsSoapHttpPort` of the Web module `helloWorldJaxws#1!helloWorldJaxws`. The web service is part of the application `helloWorldJaxws` for the server `AdminServer` in the domain `wls-domain`.

```
wls:/wls-domain/serverConfig>enableWebServicePolicies
('/wls-domain/AdminServer/helloWorldJaxws','helloWorldJaxws#1!helloWorldJaxws',
'wls','helloWorldJaxws', 'helloWorldJaxwsSoapHttpPort', ["oracle/binding_
authorization_denyall_policy", "oracle/wss_username_token_service_policy"], false)
```

## enableWebServicePolicy

---

**Note:** Use this command for Java EE Web Services (or clients) only. It has been deprecated for Oracle Infrastructure Web Services.

For Oracle Infrastructure Web Services, it is recommended that you use the `enableWSMPolicy` command, as described in "enableWSMPolicy" on page 3-85. The following examples show how to migrate to use the `enableWSMPolicy` command.

11*g* Release:

```
wls:/wls-domain/serverConfig>enableWebServicePolicy
('/base_domain/server1/HelloWorld#1_0','j2wbasicPolicy','web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort',"oracl
e/wss_username_token_service_policy",true)
```

12*c* Release:

```
wls:/wls-domain/serverConfig>enableWSMPolicy("oracle/wss_username_
token_service_policy",true)
```

---

Command Category: Policy Management

Use with WLST: Online

### Description

Enables or disables a policy attached to a port of a web service application or SOA composite.

If the policy that you specify in this command is not attached to the port, an error message is displayed and/or an exception is thrown.

---

**Note:** Policy changes made using this WLST command are only effective after you restart your application.

---

### Syntax

```
enableWebServicePolicy(application, moduleOrCompName, moduleType, serviceName,
subjectName, policyURI, [enable], [subjectType=None] ))
```

| Argument | Definition |
|----------|-----------|
| *application* | Name and path of the application for which you want to enable a web service policy. For example, /domain/server/application#version_number |
| | To enable a policy that is attached to a port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to enable a web service policy. |
| | To enable a policy that is attached to a port of a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |

| Argument | Definition |
|----------|------------|
| *moduleType* | Module type. Valid options are:<br><br>■ `soa`—SOA composite.<br><br>■ `web`—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br><br>■ `wls`—Java EE web services.<br><br>**Note**: The `web` module type is deprecated for this release. |
| *serviceName* | Name of the web service in the application or SOA composite. For example, `{http://namespace/}serviceName`. Note that the namespace (`{http://namespace/}`) should not be included for a SOA composite. |
| *subjectName* | Name of the policy subject, port, or operation. |
| *policyURI* | OWSM policy name URI, for example, `'oracle/log_policy'`<br><br>If the policy that you specify is not attached, an error message is displayed and/or an exception is thrown. |
| *enable* | Optional. Specifies whether to enable or disable the policy. Valid options are:<br><br>■ `true`—Enables the policy. The default is `true`.<br><br>■ `false`—Disables the policy.<br><br>If you omit this argument, the policy is enabled. |
| *subjectType* | Optional. Policy subject type. Valid options are:<br><br>■ `P`—Port. The default is `P`.<br><br>■ `O`—Not supported in this release. |

**Examples**

The following example enables the policy `oracle/log_policy` attached to the port `HelloWorld_pt` for the service `HelloService` in the SOA composite `default/HelloWorld[1.0]`. Note that the namespace (`{http://namespace/}`) should not be included for a SOA composite.

```
wls:/wls-domain/serverConfig>enableWebServicePolicy(None,
'default/HelloWorld[1.0]', 'soa','HelloService','HelloWorld_pt','oracle/log_
policy')
```

The following example disables the policy `oracle/log_policy` attached to the port `HelloWorld_pt` for the service `HelloService` in the SOA composite `default/HelloWorld[1.0]`. Note that the namespace (`{http://namespace/}`) should not be included for a SOA composite.

```
wls:/wls-domain/serverConfig>enableWebServicePolicy(None,
'default/HelloWorld[1.0]', 'soa','HelloService','HelloWorld_pt','oracle/log_
policy',false)
```

The following example disables the policy `oracle/wss_username_token_service_policy` attached to the port `helloWorldJaxwsSoapHttpPort` for the service `helloWorldJaxws` in the Java EE web service `wls-domain/AdminServer/helloWorldJaxws`

```
wls:/wls-domain/domainRuntime> enableWebServicePolicy
('/wls-domain/AdminServer/helloWorldJaxws','helloWorldJaxws#1!helloWorldJaxws',
'wls','helloWorldJaxws', 'helloWorldJaxwsSoapHttpPort', 'oracle/wss_username_
token_service_policy', false)
```

## **enableWSMPolicies**

> **Note:** This command applies to Oracle Infrastructure and RESTful web services. It does not apply to Java EE web services in this release.

Command Category: Policy Management

Use with WLST: Online

### Description

Within a session, enables or disables multiple policy attachments, identified by the specified URIs, that are attached to a policy subject.

You must start a session and select the policy set (`selectWSMPolicySet`) or policy subject (`selectWSMPolicySubject`) before initiating the command. However, if `enableWSMPolicies` is issued when creating or cloning a policy set, there is no need to select the policy set because it is already selected.

If the optional `enable` argument is not specified, this command enables the policy attachment by default. If the policy URIs that you specify in this command are not attached to the port, an error message is displayed and/or an exception is thrown.

### Syntax

```
enableWSMPolicies(uris,[enable=true]))
```

| Argument | Definition |
|----------|------------|
| *uris* | List of OWSM policy name URIs, for example, `["oracle/log_policy","oracle/wss_username_token_service_policy"]` |
| | If the `policyURIs` that you specify are not attached, an error message is displayed and/or an exception is thrown. |
| *enable* | Optional. Specifies whether to enable or disable the policy attachments. Valid options are: |
| | ■    `true`—Enables the specified policy attachments. The default is `true`. |
| | ■    `false`—Disables the specified policy attachments. |
| | If you omit this argument, the policies are enabled. |

### Examples

The following example enables the policies `["oracle/log_policy","oracle/wss_username_token_service_policy"]` attached to the port `JRFWssUsernamePort` of the Web module `WssUsernameService`. The web service is part of the application `HelloWorld#1_0` for the server `server1` in the domain `base_domain`.

```
wls:/wls-domain/serverConfig>enableWSMPolicies(["oracle/log_policy","oracle/wss_
username_token_service_policy"],true)
```

## enableWSMPolicy

> **Note:** This command applies to Oracle Infrastructure and RESTful web services. It does not apply to Java EE web services in this release.

Command Category: Policy Management

Use with WLST: Online

### Description

Within a session, enables or disables a policy attachment, identified by a specified URI, that is attached to a policy subject.

You must start a session and select the policy set (`selectWSMPolicySet`) or policy subject (`selectWSMPolicySubject`) before initiating the command. However, if `enableWSMPolicy` is issued when creating or cloning a policy set, there is no need to select the policy set because it is already selected.

If the optional `enable` argument is not specified, this command enables the policy attachment by default. If the policyURIs that you specify in this command are not attached to the port, an error message is displayed and/or an exception is thrown.

### Syntax

```
enableWSMPolicy(uri,[enable=true])
```

| Argument | Definition |
|----------|------------|
| *uri* | URI specifying the policy attachment within the policy set. |
| *enable* | Optional. Specifies whether to enable or disable the policy attachment specified by the URI in the policy set. Valid options are: <br><br> ■ `true`—Enables the specified policy attachment in the policy set. The default is `true`. <br><br> ■ `false`—Disables specified policy attachment in the policy set. <br><br> If you omit this argument, the policy set attachment is enabled. |

### Examples

The following example enables the policy `oracle/wss_username_token_service_policy` attached to the port `JRFWssUsernamePort` of the Web module `WssUsernameService`. The web service is part of the application `HelloWorld#1_0` for the server `server1` in the domain `base_domain`.

```
wls:/wls-domain/serverConfig>enableWSMPolicy("oracle/wss_username_token_service_policy",true)
```

The following example enables the policy `oracle/log_policy` attached to the port `HelloWorld_pt` for the service `HelloService` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>enableWSMPolicy('oracle/log_policy')
```

The following example disables the policy `oracle/log_policy` attached to the port `HelloWorld_pt` for the service `HelloService` in the SOA composite `default/HelloWorld[1.0]`.

```
wls:/wls-domain/serverConfig>enableWSMPolicy('oracle/log_policy',false)
```

## listAvailableWebServicePolicies

Command Category: Policy Management

Use with WLST: Online

### Description

Displays a list of all the available OWSM policies by category or subject type.

### Syntax

```
listAvailableWebServicePolicies([category],[subject])
```

| Argument | Definition |
|----------|------------|
| *category* | Optional. The policy category, for example,: `'security'`,`'management'`. |
| *subject* | Optional. The policy subject type, for example,: `'server'` or `'client'`. |

### Example

The following example lists all the available OWSM server security policies in the domain.

```
wls:/wls-domain/serverConfig>listAvailableWebServicePolicies('security','server')
```

## listWebServiceClientPolicies

Command Category: Policy Management

Use with WLST: Online

### Description

Lists web service client port policies information for an application or SOA composite.

The output will display the web service client/reference port name, the OWSM policies it has attached to it and details about each attachment such as the policy category, status, the source of the policy attachment, any policy override properties (if applicable), and if the policy is in effect for the subject. It also displays if the policy subject is secure. For example:

```
test-port:
URI=oracle/wss_username_token_client_policy, category=security,
policy-status=enabled
 source=local policy set; reference-status=enabled; effective=true
 The policy subject is secure in this context.
```

### Syntax

```
listWebServiceClientPolicies(application, moduleOrCompName, moduleType,
serviceRefName,portInfoName)
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to list the web service client port policy information. For example, `/domain/server/application#version_number`<br><br>To list the client port policy information for a web services application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to list the web services port policy information.<br><br>To list the client port policy information for a SOA composite, the composite name is required (for example, `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`. |
| *moduleType* | Module type. Valid options are:<br>■  `soa`—SOA composite.<br>■  `web`—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br>■  `wls`—Java EE web services.<br>■  `wsconn`—Use with a connection-based web service client such as an ADF DC web service client, ADF JAX-WS Indirection Proxy. |
| *serviceRefName* | The service reference name of the application or composite. |
| *portInfoName* | The client port name. |

### Example

The following example lists the web service client port policy information for the application `jwsclient_1#1.1.0` for the server `server1` in the domain `base_domain`. In

this example, the Web module name is WssUsernameClient, the module type is wsconn, the service reference name is WssUsernameClient, and the client port name is JRFWssUsernamePort.

```
wls:/wls-domain/serverConfig>listWebServiceClientPolicies
('/base_domain/server1/jwsclient_1#1.1.0','WssUsernameClient','wsconn',
'WssUsernameClient','JRFWssUsernamePort')
```

## listWebServicePolicies

Command Category: Policy Management

Use with WLST: Online

### Description

Lists web service policy information for a web service port in an application or SOA composite.

The output will display the web service port name, the OWSM policies it has attached to it and details about each attachment such as the policy category, status, the source of the policy attachment, any policy override properties (if applicable), and if the policy is in effect for the subject. It also displays if the policy subject is secure. For example:

```
CalculatorPort:
URI="oracle/wss_username_token_service_policy", category=security,
policy-status=enabled;
 source=local policy set; reference-status=enabled; effective=true
 The policy subject is secure in this context.
```

### Syntax

```
listWebServicePolicies(application,moduleOrCompName,moduleType,serviceName,subject
Name)
```

| Argument | Definition |
|---|---|
| `application` | Name and path of the application for which you want to list the web services port policy information. For example, `/domain/server/application#version_number` |
| | To list the port policy information for a web service application, this argument is required. |
| `moduleOrCompName` | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to list the web services port policy information. |
| | To list the port policy information for a SOA composite, the composite name is required (for example, `default/HelloWorld[1.0]`), and the `moduleType` argument must be set to `soa`. |
| `moduleType` | Module type. Valid options are: |
| | ■ `soa`—SOA composite. |
| | ■ `web`—Oracle Infrastructure web services packaged as a Web module (including an EJB). |
| | ■ `wls`—Java EE web services. |
| `serviceName` | Name of the web service in the application or SOA composite for which you want to list the port policy information. For example, `{http://namespace/}serviceName`. Note that the namespace (`{http://namespace/}`) should not be included for a SOA composite. |
| `subjectName` | Policy subject, port, or operation name. |

## Examples

The following example lists the web service policy information for the port CalculatorPort in the application jaxwsejb30ws. In this example, the Web module name is jaxwsejb, and the service name is CalculatorService.

```
wls:/wls-domain/serverConfig>listWebServicePolicies ('/base_
domain/AdminServer/jaxwsejb30ws','jaxwsejb','web',
'{http://namespace/}CalculatorService', 'CalculatorPort')
```

The following example lists the port policy information for the SOA composite default/HelloWorld[1.0]. Note that the moduleType is set to SOA, the service name is HelloService, and the subject is a port named HelloWorld_pt. Note that the namespace ({http://namespace/}) should not be included for a SOA composite.

```
wls:/wls-domain/serverConfig>listWebServicePolicies (None,
'default/HelloWorld[1.0]', 'soa', 'HelloService', 'HelloWorld_pt')
```

## setWebServicePolicyOverride

> **Note:** This command has been deprecated for Oracle Infrastructure Web Services. It is recommended that you use the setWSMPolicyOverride command, as described in "setWSMPolicyOverride" on page 3-94.
>
> This command does not apply to Java EE web services.
>
> The following examples show how to migrate to use the setWSMPolicyOverride command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> setWebServicePolicyOverride
> ('/base_domain/server1/HelloWorld#1_0','j2wbasicPolicy', 'web',
> '{http://namespace/}WssUsernameService','JRFWssUsernamePort',
> 'oracle/wss_username_token_service_policy', 'reference.priority',
> '10')
> ```
>
> 12*c* Release (for repository and policy subject operations):
>
> ```
> wls:/jrfServer_domain/serverConfig> setWSMPolicyOverride
> ('oracle/wss_username_token_service_policy', 'reference.priority',
> '10')
> ```

Command Category: Policy Management

Use with WLST: Online

### Description

Configures the web service port policy override properties of an application or SOA composite.

### Syntax

```
setWebServicePolicyOverride(application,moduleOrCompName,moduleType, serviceName,
portName,policyURI,properties)
```

| Argument | Definition |
|---|---|
| *application* | Name and path of the application for which you want to override the web service port policy. For example, /domain/server/application#version_number |
| | To override properties on a policy attached to a port of a web service application, this argument is required. |
| *moduleOrCompName* | Name of the Web module or SOA composite (for example, HelloWorld[1.0]) for which you want to override a web service port policy. |
| | To override properties on a policy attached to a SOA composite, the composite name is required (for example, default/HelloWorld[1.0]), and the moduleType argument must be set to soa. |

| Argument | Definition |
|---|---|
| *moduleType* | Module type. The valid option is web—Oracle Infrastructure web services packaged as a Web module (including an EJB).<br><br>**Note:** The module type wls is not supported. |
| *serviceName* | Name of the web service in the application or SOA composite. For example, {http://namespace/}serviceName. Note that the namespace ({http://namespace/}) should not be included for a SOA composite. |
| *subjectName* | Name of the policy subject, port, or operation. |
| *policyURI* | OWSM policy name URI, for example, 'oracle/log_policy' to which the override properties will be applied.<br><br>If the policy specified is not attached, an error message is displayed and/or an exception is thrown. |
| *properties* | Policy override properties. Properties must be specified using the following format:<br><br>[("name","value")]<br><br>For example: [("myprop","myval")]<br><br>If this argument is set to None, then all policy overrides are removed. |

## Examples

The following example configures the override properties for the policy oracle/wss10_message_protection_service_policy for the port JRFWssUsernamePort of the Web module WssUsernameService. The web service is part of the application HelloWorld#1_0 for the server server1 in the domain base_domain.

```
wls:/wls-domain/serverConfig>setWebServicePolicyOverride ('/base_
domain/server1/HelloWorld#1_0','j2wbasicPolicy', 'web',
'{http://namespace/}WssUsernameService','JRFWssUsernamePort', "oracle/wss10_
message_protection_service_policy", [("keystore.sig.csf.key","sigkey")])
```

## setWSMPolicyOverride

> **Note:** For direct policy attachments, this command applies to Oracle
> Infrastructure and RESTful web services only. For configuration
> overrides on policy references within a policy set, this command also
> applies to Java EE web services. For more information about
> configuration overrides in policy sets, see "Overriding Configuration
> Properties for Globally Attached Policies Using WLST" in *Securing
> Web Services and Managing Policies with Oracle Web Services Manager*.
>
> The `local.policy.reference.source` property is for informational
> purposes only, to identify the source of the direct policy attachment,
> and should not be overridden. For more information, see
> "Determining the Source of Policy Attachments" in *Securing Web
> Services and Managing Policies with Oracle Web Services Manager*.

Command Category: Policy Management

Use with WLST: Online

### Description

Within a session, adds a configuration override, described by a `name-value` pair, to a
policy identified by the specified URI and attached to the policy set document or
policy subject. The `value` argument is optional. If the `value` argument is omitted, the
property specified by the `name` argument is removed from the policy subject. If the
property specified by the `name` argument already exists and a `value` argument is
provided, the current value is overwritten by the new value.

You must start a session and select the policy set (`selectWSMPolicySet`) or policy
subject (`selectWSMPolicySubject`) before initiating the command. If there is no
current session and no policy subject selected, an error is displayed.

### Syntax

```
setWSMPolicyOverride(uri, name, value)
```

| Argument | Description |
|----------|-------------|
| *uri* | String representing the policy URI. For example, `'oracle/wss10_saml_token_service_policy'`, to which the override properties will be applied. |
| *name* | String representing the name of the override property. For example: `['reference.priority']` |
| *value* | Optional. String representing the value of the property. If this argument is not specified, the property specified by the `name` argument, if it exists, is removed. |

### Examples

The following example specifies a configuration override for the `reference.priority`
property for the `oracle/wss10_saml_token_service_policy` to a value of `1`.

```
wls:/wls-domain/serverConfig> setWSMPolicyOverride('oracle/wss10_saml_token_
service_policy', 'reference.priority','1')
```

The following example removes the property `reference.priority` from the `oracle/wss10_saml_token_service_policy` in the policy set.

```
wls:/wls-domain/serverConfig> setWSMPolicyOverride('oracle/wss10_saml_token_
service_policy', 'reference.priority')
```

# Policy Set Management Commands

Policy sets enhance the security and manageability of an enterprise by providing a mechanism to globally attach one or more policies to a subject type. Using policy sets, an administrator can specify a default set of policies to be enforced even if none are directly attached. For detailed information about determining the type and scope of resources a policy set can be attached to, see "Defining the Type and Scope of Resources for Globally Attached Policies" in the *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

All policy set creation, modification, or deletion commands must be performed in the context of a session. A session can only act on a single policy set.

> **Notes:** To view the help for the WLST commands described in this section, connect to a running instance of the server and enter `help('wsmManage')`.
>
> The policy set management commands listed in Table 3–9 have been deprecated in this release for Oracle Infrastructure Web Services.
>
> For Oracle Infrastructure Web Services, Oracle recommends that you use the new WLST commands listed in Table 3–8 to manage OWSM policy sets in release 12*c*. These commands must be executed within the context of a session using the session commands described in Section , "Session Commands."
>
> For a complete list of deprecated commands, see "Deprecated Commands for Oracle Infrastructure Web Services" in *Release Notes for Oracle Fusion Middleware Infrastructure*.

Use the WLST commands listed in Table 3–6 to manage globally available policy sets.

*Table 3–8    Web Services Global Policy Set Management WLST Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| cloneWSMPolicySet | Within a session, clone a new policy set from an existing policy set. | Online |
| createWSMPolicySet | Create a new, empty policy set within a session. | Online |
| deleteWSMAllPolicySets | Delete all or selected policy sets from within the OWSM repository. | Online |
| deleteWSMPolicySet | Delete a specified policy set within a session. | Online |
| displayWSMPolicySet | Display the configuration of a specified policy set. | Online |
| enableWSMPolicySet | Enable or disable the current policy set within a session. | Online |
| listWSMPolicySets | Lists the policy sets in the repository. This command will also display a policy set that is being created, modified, or deleted within the current session. | Online |
| selectWSMPolicySet | Specify a policy set for modification within a session. | Online |

*Table 3–8   (Cont.) Web Services Global Policy Set Management WLST Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| setWSMPolicySetConstraint | Specify a run-time constraint value for a policy set selected within a session. | Online |
| setWSMPolicySetOverride | Configure override properties to a policy set. | Online |
| setWSMPolicySetDescription | Specify a description for a policy set selected within a session. | Online |
| setWSMPolicySetScope | Set an expression that attaches a policy set to the specified resource scope. | Online |
| validateWSMPolicySet | Validate an existing policy set. | Online |

Table 3–9 list the WLST commands that are deprecated in this release for managing Oracle Infrastructure web service global policy sets.

*Table 3–9    Deprecated WLST Commands for Global Policy Set Management*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| abortRepositorySession | Abort the current OWSM repository modification session, discarding any changes that were made to the repository during the session. | Online |
| attachPolicySet | Attach a policy set to the specified resource scope. | Online |
| attachPolicySetPolicy | Attach a policy to a policy set using the policy's URI. | Online |
| beginRepositorySession | Begin a session to modify the OWSM repository. | Online |
| clonePolicySet | Clone a new policy set from an existing policy set. | Online |
| commitRepositorySession | Write the contents of the current session to the OWSM repository. | Online |
| createPolicySet | Create a new, empty policy set. | Online |
| deleteAllPolicySets | Delete all or selected policy sets from within the OWSM repository. | Online |
| deletePolicySet | Delete a specified policy set. | Online |
| describeRepositorySession | Describe the contents of the current session. | Online |
| detachPolicySetPolicy | Detach a policy from a policy set using the policy's URI. | Online |
| displayPolicySet | Display the configuration of a specified policy set. | Online |
| enablePolicySet | Enable or disable a policy set. | Online |
| enablePolicySetPolicy | Enable or disable a policy attachment for a policy set using the policy's URI. | Online |
| listPolicySets | List the policy sets in the repository. | Online |
| migrateAttachments | Migrate direct policy attachments to global policy attachments if they are identical. | Online |

*Table 3–9 (Cont.) Deprecated WLST Commands for Global Policy Set Management*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| modifyPolicySet | Specify an existing policy set for modification in the current session. | Online |
| setPolicySetConstraint | Specify a run-time constraint value for a policy set selected within a session. | Online |
| setPolicySetDescription | Specify a description for the policy set selected within a session. | Online |
| setPolicySetPolicyOverride | Add a configuration override to a policy reference in the current policy set. | Online |
| validatePolicySet | Validate an existing policy set in the repository or in a session. | Online |

## abortRepositorySession

> **Note:** This command has been deprecated. It is recommended that you use the `abortWSMSession` command, as described in "abortWSMSession" on page 3-12.
>
> The following examples show how to migrate to use the `abortWSMSession` command.
>
> 11*g* Release (for Repository operations):
>
> ```
> wls:/jrfServer_domain/serverConfig> abortRepositorySession()
> ```
>
> 12*c* Release (for both Repository and PolicySubject operations):
>
> ```
> wls:/jrfServer_domain/serverConfig> abortWSMSession()
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Aborts the current modification session, discarding any changes that were made to the repository during the session.

### Syntax

```
abortRepositorySession()
```

### Example

The following example aborts the current OWSM session.

```
wls:/wls-domain/serverConfig>abortRepositorySession()
```

## **attachPolicySet**

> **Note:** This command has been deprecated. It is recommended that you use the `setWSMPolicySetScope` command, as described in "setWSMPolicySetScope" on page 3-137.
>
> The following examples show how to migrate to use the `setWSMPolicySetScope` command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> attachPolicySet ('Domain("base_
> domain")')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> setWSMPolicySetScope
> ('Domain("base_domain")')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, sets an expression that attaches a policy set to the specified resource scope. The expression must define a valid resource scope in a supported format.

Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

### Syntax

```
attachPolicySet(expression)
```

| Argument | Definition |
|---|---|
| *expression* | Expression that attaches the policy set to the specified resource scope. |
| | For details about specifying the resource scope expression, see "Defining the Resource Scope" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*. |

### Example

The following example attaches a policy set to the specified `base_domain` resource.

```
wls:/wls-domain/serverConfig>attachPolicySet('Domain("base_domain")')
```

This example attaches a policy set to the specified `base_domain` **and** `managed_server` resources.

```
wls:/wls-domain/serverConfig>attachPolicySet('Domain("base_domain") and
Server("managed_server")')
```

## attachPolicySetPolicy

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the attachWSMPolicy command, as described in "attachWSMPolicy" on page 3-65. The following examples show how to migrate to use the attachWSMPolicy command.
>
> 11*g* Release (for both Repository and PolicySubject operation on policy set):
>
> wls:/jrfServer_domain/serverConfig> **attachPolicySetPolicy ('oracle/wss_username_token_service_policy')**
>
> 12*c* Release:
>
> wls:/jrfServer_domain/serverConfig> **attachWSMPolicy('oracle/wss_ username_token_service_policy')**

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, attaches a policy, identified by the specified URI, to the current policy set.

Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

### Syntax

attachPolicySetPolicy(uri)

| Argument | Definition |
|----------|------------|
| *uri* | URI specifying the policy to attach to the current policy set. For example, 'oracle/log_policy'. |

### Example

The following example attaches the OWSM logging policy to the current policy set.

wls:/wls-domain/serverConfig>**attachPolicySetPolicy('oracle/log_policy')**

## beginRepositorySession

> **Note:** This command has been deprecated. It is recommended that you use the beginWSMSession command, as described in "beginWSMSession" on page 3-13.
>
> The following examples show how to migrate to use the beginWSMSession command.
>
> 11*g* Release (for Repository operations):
>
> ```
> wls:/jrfServer_domain/serverConfig> beginRepositorySession()
> ```
>
> 12*c* Release (for both Repository and PolicySubject operations):
>
> ```
> wls:/jrfServer_domain/serverConfig> beginWSMSession()
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Begins a session to modify the OWSM Repository. A session can only act on a single policy subject, such as a policy set or a Fusion Middleware web service endpoint. An error will be displayed if there is already a current session.

### Syntax

```
beginRepositorySession()
```

### Example

The following example begins an OWSM Repository modification session.

```
wls:/wls-domain/serverConfig>beginRepositorySession()
```

## clonePolicySet

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the `cloneWSMPolicySet` command, as described in "cloneWSMPolicySet" on page 3-105. The following examples show how to migrate to use the `cloneWSMPolicySet` command.
>
> 11*g* Release:
>
> `wls:/jrfServer_domain/serverConfig>` **`clonePolicySet`**
> **`('myNewPolicySet', 'myPolicySet')`**
>
> 12*c* Release:
>
> `wls:/jrfServer_domain/serverConfig>` **`cloneWSMPolicySet`**
> **`('myNewPolicySet', 'myPolicySet')`**

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, clones a new policy set from an existing policy set. When cloning an existing policy set, all values and attachments in the source policy set are copied into the new policy set, although you can supply a different expression identifying the resource scope. The expression must define a valid resource scope in a supported format.

Issuing this command outside of a session will result in an error.

### Syntax

```
clonePolicySet(name, source,[attachTo=None],[description=None],[enable='true'])
```

| Argument | Definition |
|---|---|
| *name* | Name of the new policy set clone. |
| *source* | Name of the source policy set that will be cloned. |
| *attachTo=None* | Optional. Expression that attaches the policy set to the specified resource scope. For details about specifying the resource scope expression, see "Defining the Resource Scope" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*. |
| | If this argument is set to `None`, then the expression used in the source policy set to identify the scope of resources is retained. |
| *description=None* | Optional. Description for the new policy set. |
| | If this argument is set to `None`, then the description used in the source policy set is retained. |

| Argument | Definition |
|----------|------------|
| *enable='true'* | Optional. Specifies whether to enable or disable the policy set. Valid options are:<br><br>■    true—Enables the policy set. The default is true.<br><br>■    false—Disables the policy set.<br><br>If you omit this argument, the policy set is enabled. |

## Example

The first example creates a policy set by cloning the existing *myPolicySet* policy set to create a new *mynewPolicySet*. The second example also creates a policy set, but narrows the resource scope to policy subjects in the specified jaxwsejb30ws application in the domain.

```
wls:/wls-domain/serverConfig>clonePolicySet('myNewPolicySet','myPolicySet')
wls:/wls-domain/serverConfig>clonePolicySet('myNewPolicySet','myPolicySet','Application("jaxwsejb30ws")')
```

## cloneWSMPolicySet

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, clones a new policy set from an existing policy set. When cloning an existing policy set, all values and attachments in the source policy set are copied into the new policy set, although you can supply a different expression identifying the resource scope. The expression must define a valid resource scope in a supported format.

Issuing this command outside of a session will result in an error.

### Syntax

```
cloneWSMPolicySet(name,source,[scope=None],[description=None],[enable='true'])
```

| Argument | Definition |
| --- | --- |
| *name* | Name of the new policy set clone. |
| *source* | Name of the source policy set that will be cloned. |
| *scope=None* | Optional. Expression that attaches the policy set to the specified resource scope. For details about specifying the resource scope expression, see "Defining the Resource Scope" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*. |
| | If this argument is not specified, then the expression used in the source policy set to identify the scope of resources is retained. |
| *description=None* | Optional. Description for the new policy set. |
| | If this argument is not specified, then the description used in the source policy set is retained. |
| *enable='true'* | Optional. Specifies whether to enable or disable the policy set. If you omit this argument, the policy set is enabled. Valid options are: |
| | ■   true—Enables the policy set. The default is true. |
| | ■   false—Disables the policy set. |
| | If you omit this argument, the policy set is enabled. |

### Examples

The first example creates a policy set by cloning the existing *myPolicySet* policy set to create a new *mynewPolicySet*. The second example also creates a policy set, but narrows the resource scope to policy subjects in the specified jaxwsejb30ws application in the domain.

```
wls:/wls-domain/serverConfig>cloneWSMPolicySet('myNewPolicySet','myPolicySet')
wls:/wls-domain/serverConfig>cloneWSMPolicySet('myNewPolicySet','myPolicySet','Application("jaxwsejb30ws")')
```

## commitRepositorySession

> **Note:**  This command has been deprecated. It is recommended that you use the `commitWSMSession` command, as described in "commitWSMSession" on page 3-14.
>
> The following examples show how to migrate to use the `commitWSMSession` command.
>
> 11*g* Release (for Repository operations):
>
> ```
> wls:/jrfServer_domain/serverConfig> commitRepositorySession()
> ```
>
> 12*c* Release (for both Repository and PolicySubject operations):
>
> ```
> wls:/jrfServer_domain/serverConfig> commitWSMSession()
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Writes the contents of the current session to the OWSM 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()
```

## createPolicySet

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the createWSMPolicySet command, as described in "createWSMPolicySet" on page 3-109. The following examples show how to migrate to use the createWSMPolicySet command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> createPolicySet('myPolicySet',
> 'ws-service', 'Domain("base_domain")')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> createWSMPolicySet
> ('myPolicySet', 'ws-service', 'Domain("base_domain")')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Creates a new, empty policy set within a session. When creating a new policy set, you must specify the type of policy subject that the policy set will apply to, and a supported expression that defines a valid resource scope in a supported format.

Issuing this command outside of a session will result in an error.

### Syntax

```
createPolicySet(name,type,attachTo,[description=None],[enable='true'])
```

| Argument | Definition |
|----------|------------|
| *name* | Name of the new, empty policy set. |
| *type* | The type of policy subject to which the new policy set applies. The type of policy subject must be one of the policy subjects described in "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*. |
| *attachTo* | Expression that attaches the policy set to the specified resource scope. For details about specifying the resource scope expression, see "Defining the Resource Scope" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*. |
| *description* | Optional. Description of the new policy set. If no description is specified, then the description for a new policy set will be "Global policy attachments for *<type>*", where *<type>* is the subject type. |
| *enable* | Optional. Specifies whether to enable or disable the new policy set. Valid options are:<br><br>■   true—Enables the new policy set. The default is true.<br><br>■   false—Disables the new policy set.<br><br>If you omit this argument, the policy set is enabled. |

## Example

The first example creates a new policy set and specifies the resource scope to only ws-service types (Web Service Endpoint) in the base_domain domain. The second example creates a new policy set, but also narrows the resource scope to only sca-service types (SOA Service) in the soa_server1 server in the domain.

```
wls:/wls-domain/serverConfig>createPolicySet('myPolicySet','ws-service','Domain("b
ase_domain")')
wls:/wls-domain/serverConfig>createPolicySet('myPolicySet','sca-service','Server("
soa_server1")','My policySet')
```

## createWSMPolicySet

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, creates a new, empty policy set. When creating a new policy set, you must specify the type of policy subject that the policy set will apply to, and provide a supported expression that defines a valid resource scope in a supported format.

Issuing this command outside of a session will result in an error.

### Syntax

```
createWSMPolicySet(name,type,scope,[description=None],[enable='true'])
```

| Argument | Definition |
|---|---|
| *name* | Name of the new, empty policy set. |
| *type* | The type of policy subject that the new policy set applies to. The type of policy subject must be one of the policy subjects described in "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*. |
| *scope* | Optional. Expression that attaches the policy set to the specified resource scope. For details about specifying the resource scope expression, see "Defining the Resource Scope" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*. |
| | If this argument is not specified, then the expression used in the source policy set to identify the scope of resources is retained. |
| *description=None* | Optional. Description of the new policy set. If no description is specified, then the description for a new policy set will be "Global policy attachments for *<type>*", where *<type>* is the subject type. |
| *enable='true'* | Optional. Specifies whether to enable or disable the new policy set. Valid options are: |
| | ■    true—Enables the new policy set. The default is true. |
| | ■    false—Disables the new policy set. |
| | If you omit this argument, the policy set is enabled. |

### Examples

The following example creates a new policy set and specifies the resource scope to only ws-service types (Web Service Endpoint) in the base_domain domain.

```
wls:/wls-domain/serverConfig>createWSMPolicySet('myPolicySet','ws-service','Domain
("base_domain")')
```

The following example creates a new policy set, but also narrows the resource scope to only sca-service types (SOA Service) in the soa_server1 server in the domain.

```
wls:/wls-domain/serverConfig>createWSMPolicySet('myPolicySet','sca-service','Serve
r("soa_server1")','My policySet')
```

The following example creates a new policy set, narrowing the resource scope to only sca-rest-reference types (SOA RESTful references) in the base_domain domain.

```
wls:/wls-domain/serverConfig>createWSMPolicySet('myPolicySet','sca-rest-reference'
,'Domain("base_domain")','My policySet')
```

The following example creates a new policy set, narrowing the resource scope to only sca-rest-reference types (OSB RESTful business services) in the base_domain domain.

```
wls:/wls-domain/serverConfig>createWSMPolicySet('myPolicySet','biz-rest-service','
Domain("base_domain")','My policySet')
```

## deleteAllPolicySets

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the deleteWSMAllPolicySets command, as described in "deleteWSMAllPolicySets" on page 3-113. The following examples show how to migrate to use the deleteWSMAllPolicySets command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> deleteAllPolicySets()
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> deleteWSMAllPolicySets()
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Deletes all or selected policy sets from within the OWSM repository. You can specify whether to force deletion of all the policy sets, or prompt to select individual policy sets for deletion. If deletion of any policy set fails then this operation throws an exception and no policy sets are deleted.

### Syntax

```
deleteAllPolicySets([mode])
```

| Argument | Definition |
|----------|------------|
| *mode* | Optional. The action to be taken for performing policy set deletion. Valid options are: |
| | ■ force—Automatically delete all policy sets without prompting. |
| | ■ prompt—Request user confirmation for each policy set deletion. Available options are yes, no, and cancel. If you select cancel for any property set deletion, the operation is canceled and no policy sets are deleted. |
| | If no mode is specified, this argument defaults to prompt mode. |

### Examples

The following example automatically deletes all policy sets from the respository without prompting.

```
wls:/jrfServer_domain/serverConfig> deleteAllPolicySets("force")

Starting Operation deleteAllPolicySets ...

All policy sets were deleted successfully from repository.

deleteAllPolicySets Operation Completed.
```

The following examples delete selected policy sets from the repository.

```
wls:/jrfServer_domain/serverConfig> deleteAllPolicySets()
```

or

wls:/jrfServer_domain/serverConfig> **deleteAllPolicySets('prompt')**

Starting Operation deleteAllPolicySets ...

Policy Set Name: create_policyset_6
Select "create_policyset_6" for deletion (yes/no/cancel)? no
Policy Set Name: create_policyset_8
Select "create_policyset_8" for deletion (yes/no/cancel)? yes
Policy Set Name: create_policyset_21
Select "create_policyset_21" for deletion (yes/no/cancel)? no
Policy Set Name: create_policyset_10
Select "create_policyset_10" for deletion (yes/no/cancel)? yes

All the selected policy sets were deleted successfully from repository.

deleteAllPolicySets Operation Completed.

## deleteWSMAllPolicySets

Command Category: Policy Set Management

Use with WLST: Online

### Description

Deletes all or selected policy sets within a session. You can specify whether to force deletion of all the policy sets, or prompt to select individual policy sets for deletion. If deletion of any policy set fails then this operation throws an exception and no policy sets are deleted.

### Syntax

```
deleteWSMAllPolicySets([mode])
```

| Argument | Definition |
|----------|-----------|
| *mode* | Optional. The action to be taken for performing policy set deletion. Valid options are: |
|  | ■   force—Automatically delete all policy sets without prompting. |
|  | ■   prompt—Request user confirmation for each policy set deletion. Available options are yes, no, and cancel. If you select cancel for any property set deletion, the operation is canceled and no policy sets are deleted. |
|  | If no mode is specified, this argument defaults to prompt mode. |

### Examples

The following example automatically deletes all policy sets from the respository without prompting.

```
wls:/jrfServer_domain/serverConfig> deleteWSMAllPolicySets("force")

Starting Operation deleteWSMAllPolicySets ...

All policy sets were deleted successfully from repository.

deleteWSMAllPolicySets Operation Completed.
```

The following examples delete selected policy sets from the repository.

```
wls:/jrfServer_domain/serverConfig> deleteWSMAllPolicySets()
```

or

```
wls:/jrfServer_domain/serverConfig> deleteWSMAllPolicySets('prompt')

Starting Operation deleteWSMAllPolicySets ...

Policy Set Name: create_policyset_6
Select "create_policyset_6" for deletion (yes/no/cancel)? no
Policy Set Name: create_policyset_8
Select "create_policyset_8" for deletion (yes/no/cancel)? yes
Policy Set Name: create_policyset_21
Select "create_policyset_21" for deletion (yes/no/cancel)? no
Policy Set Name: create_policyset_10
Select "create_policyset_10" for deletion (yes/no/cancel)? yes
```

All the selected policy sets were deleted successfully from repository.

deleteWSMAllPolicySets Operation Completed.

## deletePolicySet

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the deleteWSMPolicySet command, as described in "deleteWSMPolicySet" on page 3-116. The following examples show how to migrate to use the deleteWSMPolicySet command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> deletePolicySet('myPolicySet')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> deleteWSMPolicySet
> ('myPolicySet')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Deletes a specified policy set within a session. If the session already contains a different policy set, an error will display. If the session already contains the named policy set, then a creation will be undone or a modification will be converted into a deletion.

Issuing this command outside of a session will result in an error.

### Syntax

```
deletePolicySet(name)
```

| Argument | Definition |
|----------|------------|
| *name* | Name of the policy set to be deleted. |

### Example

The following example deletes a specified *myPolicySet* policy set.

```
wls:/wls-domain/serverConfig>deletePolicySet('myPolicySet')
```

## deleteWSMPolicySet

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, deletes a specified policy set. If the session already contains a different policy set, an error will display. If the session already contains the named policy set, then a creation will be undone or a modification will be converted into a deletion.

Issuing this command outside of a session will result in an error.

### Syntax

deleteWSMPolicySet(name)

| Argument | Definition |
|----------|------------|
| *name* | Name of the policy set to be deleted. |

### Examples

The following example deletes a specified *myPolicySet* policy set.

wls:/wls-domain/serverConfig>**deleteWSMPolicySet('myPolicySet')**

## describeRepositorySession

> **Note:** This command has been deprecated. It is recommended that you use the describeWSMSession command, as described in "describeWSMSession" on page 3-15. The following examples show how to migrate to use the describeWSMSession command.
>
> 11*g* Release (for Repository operations):
>
> ```
> wls:/jrfServer_domain/serverConfig> describeRepositorySession()
> ```
>
> 12*c* Release (for both Repository and Policy Subject operations):
>
> ```
> wls:/jrfServer_domain/serverConfig> describeWSMSession()
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Describes the contents of the current session. This will either indicate that the session is empty or list the name of the policy subject 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()
```

### Example

The following example describes the current repository modification session.

```
wls:/wls-domain/serverConfig>describeRepositorySession()
```

## detachPolicySetPolicy

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the detachWSMPolicy command, as described in "detachWSMPolicy" on page 3-75. The following examples show how to migrate to use the detachWSMPolicy command.
>
> 11*g* Release (for both Repository and Policy Subject operations on policy set):
>
> ```
> wls:/jrfServer_domain/serverConfig> detachPolicySetPolicy
> ('oracle/wss_username_token_service_policy')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> detachWSMPolicy('oracle/wss_
> username_token_service_policy')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, detaches a policy, identified by a specified URI, from the current policy set.

Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

### Syntax

```
detachPolicySetPolicy(uri)
```

| Argument | Definition |
|----------|------------|
| *uri* | URI specifying the policy to detach to the current policy set. For example, oracle/log_policy'. |

### Example

The following example detaches the OWSM logging policy from the current policy set.

```
wls:/wls-domain/serverConfig> detachPolicySetPolicy('oracle/log_policy')
```

## displayPolicySet

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the displayWSMPolicySet command, as described in "displayWSMPolicySet" on page 3-120. The following examples show how to migrate to use the displayWSMPolicySet command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> displayPolicySet('myPolicySet')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> displayWSMPolicySet
> ('myPolicySet')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Displays the configuration of a specified policy set. If the policy set is being modified in the current session, then that version will be displayed; otherwise, the latest version in the repository will be displayed. An error will display if the policy set does not exist.

This command can be issued outside of a session.

### Syntax

```
displayPolicySet([name])
```

| Argument | Definition |
|----------|------------|
| *name* | Optional. Name of the policy set to be displayed. |
|  | If a name is not specified, the configuration of the policy set, if any, in the current session is displayed or an error message is displayed. |

### Example

The following example displays the configuration of the *myPolicySet* policy set.

```
wls:/wls-domain/serverConfig>displayPolicySet('myPolicySet')
```

## displayWSMPolicySet

Command Category: Policy Set Management

Use with WLST: Online

### Description

Displays the configuration of a specified policy set. If the policy set is being modified in the current session, then that version will be displayed; otherwise, the latest version in the repository will be displayed. An error will display if the policy set does not exist.

This command can be issued outside of a session.

### Syntax

```
displayWSMPolicySet([name])
```

| Argument | Definition |
|----------|------------|
| *name* | Optional. Name of the policy set to be displayed. |
| | If a name is not specified, the configuration of the policy set, if any, in the current session is displayed or an error message is displayed. |

### Examples

The following example displays the configuration of the *myPolicySet* policy set.

```
wls:/wls-domain/serverConfig>displayWSMPolicySet('myPolicySet')
```

## enablePolicySet

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the `enableWSMPolicySet` command, as described in "enableWSMPolicySet" on page 3-123. The following examples show how to migrate to use the `enableWSMPolicySet` command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> enablePolicySet(true)
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> enableWSMPolicySet(true)
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Enables or disables the current policy set within a session. If not specified, this command enables the policy set.

Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

### Syntax

```
enablePolicySet([enable=True])
```

| Argument | Definition |
|----------|------------|
| *enable* | Optional. Specifies whether to enable or disable the policy set. Valid options are: <br><br> ■   `true`—Enables the policy set. The default is `true`. <br><br> ■   `false`—Disables the policy set. <br><br> If you omit this argument, the policy set is enabled. |

### Example

The following example enables the current policy set.

```
wls:/wls-domain/serverConfig>enablePolicySet(true)
```

## enablePolicySetPolicy

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the `enableWSMPolicySet` command, as described in "enableWSMPolicySet" on page 3-123. The following examples show how to migrate to use the `enableWSMPolicySet` command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>enablePolicySetPolicy('/oracle/log_
> policy',false)
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig>enableWSMPolicy('/oracle/log_
> policy',false)
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, enables or disables the policy attachment, which is identified by the provided URI in the current policy set. If not specified, this command enables the policy set. An error displays if the identified policy is not currently attached to the policy set.

Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

### Syntax

```
enablePolicySetPolicy(uri,[enable=true])
```

| Argument | Definition |
|----------|-----------|
| *uri* | URI specifying the policy attachment within the policy set. |
| *enable* | Optional. Specifies whether to enable or disable the policy attachment specified by the URI in the policy set. Valid options are:<br><br>■ `true`—Enables the specified policy attachment in the policy set. The default is `true`.<br><br>■ `false`—Disables specified policy attachment in the policy set.<br><br>If you omit this argument, the policy set attachment is enabled. |

### Example

The following example disables the specified logging policy attachment within the current policy set.

```
wls:/wls-domain/serverConfig>enablePolicySetPolicy('/oracle/log_policy',false)
```

## enableWSMPolicySet

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, enables or disables the current policy set. If the optional `enable` argument is not specified, this command enables the policy set by default.

Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

### Syntax

```
enableWSMPolicySet([enable=True])
```

| Argument | Definition |
|----------|------------|
| *enable* | Optional. Specifies whether to enable or disable the policy set. Valid options are: |
| | ■ `true`—Enables the policy set. The default is `true`. |
| | ■ `false`—Disables the policy set. |
| | If you omit this argument, the policy set is enabled. |

### Examples

The following example enables the current policy set.

```
wls:/wls-domain/serverConfig>enableWSMPolicySet(true)
```

## listPolicySets

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the `listWSMPolicySets` command, as described in "listWSMPolicySets" on page 3-125. The following examples show how to migrate to use the `listWSMPolicySets` command.
>
> 11*g* Release:
>
> ```
> wls:/wls-domain/serverConfig>listPolicySets('ws-service')
> ```
>
> 12*c* Release:
>
> ```
> wls:/wls-domain/serverConfig>listWSMPolicySets('ws-service')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Lists the policy sets in the repository. This command will also display a policy set that is being created, modified, or deleted within the current session. You can list all the policy sets or limit the display to include only those that apply to specific policy subject resource types.

### Syntax

```
listPolicySets([type=None])
```

| Argument | Definition |
|---|---|
| *type=None* | Optional. Specifies the type of policy subject for which the associated policy sets will be displayed. The type of policy subject must be one of the policy subjects described in "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager* |
| | If this argument is set to `None`, then all the policy sets stored in the repository will be listed. |

### Example

The first two examples list policy sets by either the `ws-service` or `ws-client` resource types. The third example lists all the policy sets stored in the repository.

```
wls:/wls-domain/serverConfig>listPolicySets('ws-service')
wls:/wls-domain/serverConfig>listPolicySets('ws-client')
wls:/wls-domain/serverConfig>listPolicySets()
```

## listWSMPolicySets

Command Category: Policy Set Management

Use with WLST: Online

### Description

Lists the policy sets in the repository. This command will also display a policy set that is being created, modified, or deleted within the current session. You can list all the policy sets or use the type argument to limit the display to include only those sets that apply to specific policy subject resource types.

### Syntax

```
listWSMPolicySets([type=None])
```

| Argument | Definition |
|---|---|
| *type=None* | Optional. Specifies the type of policy subject for which the associated policy sets will be displayed. The type of policy subject must be one of the policy subjects described in "Understanding Policy Subjects" in *Understanding Oracle Web Services Manager*. |
| | If this argument is set to None, then all the policy sets stored in the repository will be listed. |

### Examples

The first two examples list policy sets by either the ws-service or ws-client resource types. Whereas, the third example lists all the policy sets stored in the repository.

```
wls:/wls-domain/serverConfig>listWSMPolicySets('ws-service')
wls:/wls-domain/serverConfig>listWSMPolicySets('ws-client')
wls:/wls-domain/serverConfig>listWSMPolicySets()
```

## migrateAttachments

> **Note:** This command has been deprecated. It is recommended that you use the `migrateWSMAttachments` command, as described in "migrateWSMAttachments" on page 3-153. The following examples show how to migrate to use the `migrateWSMAttachments` command.
>
> 11*g* Release:
>
> `wls:/jrfServer_domain/serverConfig>` **`migrateAttachments()`**
>
> 12*c* Release:
>
> `wls:/jrfServer_domain/serverConfig>` **`migrateWSMAttachments()`**

Command Category: Policy Set Management

Use with WLST: Online

### Description

Migrates direct (local) policy attachments that are identical to the external global policy attachments that would otherwise be attached to each policy subject in the current domain. You can specify whether to force the migration, prompt for confirmation before each migration, or simply list the migrations that would occur. A direct policy attachment is identical if its URI is the same as one provided by a global policy attachment, and if it does not have any scoped configuration overrides.

> **Note:** A direct attachment with an unscoped override will be migrated but an attachment with a scoped override will not. This is because after running the `migrateAttachments()` command, the enforcement of the policies on *all* subjects remains the same, even though some policies are globally attached.

Whether forced or prompted, the command lists each direct policy attachment that is migrated. This output will identify the policy subject that was modified, the URI of the identical policy reference, and the name of the global policy attachment document that duplicated the direct attachment.

### Syntax

`migrateAttachments([mode])`

| Argument | Definition |
|----------|------------|
| *mode* | The action to be taken for each policy attachment that can be migrated. Valid options are: <br><br> ■ `force`—Automatically migrate all identical policy attachments without prompting. <br><br> ■ `preview`—List all policy attachments that can be migrated, but does not perform any migration. <br><br> ■ `prompt`—Request user confirmation before migrating each policy attachment. <br><br> If no mode is specified, this argument defaults to `prompt` mode. |

**Example**

The following examples describe how to use the repository attachment migration modes.

```
wls:/wls-domain/serverConfig>migrateAttachments()
wls:/wls-domain/serverConfig>migrateAttachments('force')
wls:/wls-domain/serverConfig>migrateAttachments('preview')
wls:/wls-domain/serverConfig>migrateAttachments('prompt')
```

## modifyPolicySet

> **Note:** For Oracle Infrastructure Web Services, it is recommended that you use the selectWSMPolicySet command, as described in "selectWSMPolicySet" on page 3-129. The following examples show how to migrate to use the selectWSMPolicySet command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> modifyPolicySet('myPolicySet')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> selectWSMPolicySet
> ('myPolicySet')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Specifies a policy set for modification in the current session. The latest version of the named policy set will be loaded into the current session. If the session already contains a different policy set, then an error will be displayed; if the session already contains the named policy set, then no action will be taken. Subsequent attempts to modify the named policy set will show the current version in the session.

Issuing this command outside of a session will result in an error.

### Syntax

```
modifyPolicySet(name)
```

| Argument | Definition |
|----------|------------|
| *name* | Name of the policy set to be modified in the current session. |

### Example

The following example opens the *myPolicySet* policy set for modification in the current session.

```
wls:/wls-domain/serverConfig>modifyPolicySet('myPolicySet')
```

## selectWSMPolicySet

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, specifies a policy set for modification. The latest version of the named policy set is loaded into the current session. If the session already contains a different policy set, then an error will be displayed; if the session already contains the named policy set, then no action will be taken. Subsequent attempts to modify the named policy set will show the current version in the session.

Issuing this command outside of a session will result in an error.

### Syntax

```
selectWSMPolicySet(name)
```

| Argument | Description |
|----------|-------------|
| *name* | Name of the policy set to be modified in the current session. |

### Examples

The following example selects a policy set in the current session named `myPolicySet`.

```
wls:/wls-domain/serverConfig> selectWSMPolicySet('myPolicySet')
```

## setPolicySetConstraint

> **Note:** This command has been deprecated. It is recommended that you use the `setWSMPolicySetConstraint` command, as described in "setWSMPolicySetConstraint" on page 3-134. The following examples show how to migrate to use the `setWSMPolicySetConstraint` command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> setPolicySetConstraint
> ('HTTPHeader("VIRTUAL_HOST_TYPE","external")')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> setWSMPolicySetConstraint
> ('HTTPHeader("VIRTUAL_HOST_TYPE","external")')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Specifies a run-time constraint value for a policy set selected within a session. Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

For more information, see "Specifying Run-time Constraints in Policy Sets" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

### Syntax

```
setPolicySetConstraint(constraint)
```

| Argument | Definition |
|----------|------------|
| *constraint* | Expression that specifies the run-time context to which the policy set applies. If not specified, the policy set applies to all run-time contexts. |

### Example

The following example specifies that the policy set apply only to requests from external clients.

```
wls:/wls-domain/serverConfig> setPolicySetConstraint('HTTPHeader("VIRTUAL_HOST_
TYPE","external")')
```

The following example specifies that the policy set apply only to requests from non-external clients.

```
wls:/wls-domain/serverConfig> setPolicySetConstraint('!HTTPHeader("VIRTUAL_HOST_
TYPE","external")')
```

## setPolicySetDescription

> **Note:** This command has been deprecated. It is recommended that you use the setWSMPolicySetDescription command, as described in "setWSMPolicySetDescription" on page 3-135. The following examples show how to migrate to use the setWSMPolicySetDescription command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> setPolicySetDescription
> ('Global policy set for web service endpoint.')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> setWSMPolicySetDescription
> ('Global policy set for web service endpoint.')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Specifies a description for a policy set selected within a session.

Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

### Syntax

```
setPolicySetDescription(description)
```

| Argument | Definition |
|---|---|
| *description* | Describes a policy set. |

### Example

The following example creates a description for a policy set.

```
wls:/wls-domain/serverConfig>setPolicySetDescription('PolicySetDescription')
```

## setPolicySetPolicyOverride

> **Note:** This command has been deprecated. It is recommended that you use the setWSMPolicyOverride command, as described in "setWSMPolicyOverride" on page 3-94. The following examples show how to migrate to use the setWSMPolicyOverride command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> setPolicySetPolicyOverride
> ('oracle/wss_username_token_service_policy', 'reference.priority',
> '10')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> setWSMPolicyOverride
> ('oracle/wss_username_token_service_policy', 'reference.priority',
> '10')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Adds a configuration override, described by a name, value pair, to an attached policy reference in the current policy set. The value argument is optional. If the value argument is omitted, the property specified by the name argument is removed from the policy reference in the policy set. If the property specified by the name argument already exists and a value argument is provided, the current value is overwritten by the new value specified with the value argument.

Issuing this command outside of a session containing a policy set that is being created or modified results in an error.

### Syntax

setPolicySetPolicyOverride(uri,name,[value=None])

| Argument | Definition |
|----------|------------|
| URI | String representing the OWSM policy URI, for example, 'oracle/wss10_saml_token_service_policy' to which the override properties will be applied. |
| name | String representing the name of the override property. For example: ['reference.priority'] |
| value | Optional. String representing the value of the property. If this argument is not specified, the property specified by the name argument, if it exists, is removed. |

### Example

The following example specifies a configuration override for the reference.priority property for the oracle/wss10_saml_token_service_policy to a value of 1.

```
wls:/wls-domain/serverConfig> setPolicySetPolicyOverride('oracle/wss10_saml_token_
service_policy', 'reference.priority','1')
```

The following example removes the property `reference.priority` from the
`oracle/wss10_saml_token_service_policy` in the policy set.

```
wls:/wls-domain/serverConfig> setPolicySetPolicyOverride('oracle/wss10_saml_token_
service_policy', 'reference.priority')
```

## setWSMPolicySetConstraint

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, specifies a constraint value for a policy set selected within a session. Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

For more information, see "Specifying Run-time Constraints in Policy Sets" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

### Syntax

```
setWSMPolicySetConstraint(constraint)
```

| Argument | Definition |
|----------|------------|
| *constraint* | Expression that specifies the run-time context to which the policy set applies. If not specified, the policy set applies to all run-time contexts. |

### Examples

The following example specifies that the policy set applies only to requests from external clients.

wls:/wls-domain/serverConfig> **setWSMPolicySetConstraint('HTTPHeader("VIRTUAL_HOST_TYPE","external")')**

The following example specifies that the policy set applies only to requests from non-external clients.

wls:/wls-domain/serverConfig> **setWSMPolicySetConstraint('!HTTPHeader("VIRTUAL_HOST_TYPE","external")')**

## setWSMPolicySetDescription

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, specifies a description for a policy set. Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

### Syntax

```
setWSMPolicySetDescription(description)
```

| Argument | Definition |
|----------|------------|
| *description* | Describes a policy set. |

### Examples

The following example creates a description for a policy set.

```
wls:/wls-domain/serverConfig>setWSWPolicySetDescription('PolicySetDescription')
```

## setWSMPolicySetOverride

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, adds a configuration override, described by a `name-value` pair, to the currently selected policy set. The override is unscoped to any specific policy reference. The `value` argument is optional. If the `value` argument is omitted, a null is assumed for `value`, and the property specified by the `name` argument is removed from the policy set. If the property specified by the `name` argument already exists and a `value` argument is provided, the current value is overwritten by the new value.

You must start a session and select the policy set (using the `selectWSMPolicySet` command), before initiating the command. Issuing this command outside of a session containing a policy subject that is being created or modified results in an error.

### Syntax

```
setWSMPolicySetOverride(name,[value=None])
```

| Argument | Description |
|----------|-------------|
| *name* | String representing the name of the override property. For example: `['on.behalf.of']` |
| *value* | Optional. String representing the value of the property. If this argument is not specified, a null is assumed and the property specified by the `name` argument is removed, if one exists with the same name. |

### Examples

The following example specifies a configuration override for the `on.behalf.of` property for the policy set selected in the session to a value of `true`.

```
wls:/wls-domain/serverConfig> setWSMPolicySetOverride('on.behalf.of','true')
```

The following example removes the property `on.behalf.of` from the policy set.

```
wls:/wls-domain/serverConfig> setWSMPolicySetOverride('on.behalf.of')
```

## setWSMPolicySetScope

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, sets an expression that attaches a policy set to the specified resource scope. The expression must define a valid resource scope in a supported format.

Issuing this command outside of a session containing a policy set that is being created or modified will result in an error.

### Syntax

```
setWSMPolicySetScope(expression)
```

| Argument | Definition |
|----------|------------|
| *expression* | Expression that attaches the policy set to the specified resource scope. |
| | For details about specifying the resource scope expression, see "Defining the Resource Scope" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*. |

### Examples

The following example attaches a policy set to the specified base_domain resource.

```
wls:/wls-domain/serverConfig>setWSMPolicySetScope('Domain("base_domain")')
```

This example attaches a policy set to the specified base_domain **and** managed_server resources.

```
wls:/wls-domain/serverConfig>setWSMPolicySetScope('Domain("base_domain") and
Server("managed_server")')
```

## validatePolicySet

> **Note:** This command has been deprecated. It is recommended that you use the validateWSMPolicySet command, as described in "validateWSMPolicySet" on page 3-139. The following examples show how to migrate to use the validateWSMPolicySet command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> validatePolicySet
> ('myPolicySet')
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> validateWSMPolicySet
> ('myPolicySet')
> ```

Command Category: Policy Set Management

Use with WLST: Online

### Description

Validates an existing policy set. If a policy set name is provided, the command will validate the specified policy set. If no policy set name is specified, the command will validate the policy set in the current session.

An error message displays if the policy set does not exist, or a name is not provided and the session is not active, or if the OWSM repository does not contain a suitable policy set.

### Syntax

```
validatePolicySet([name=None])
```

| Argument | Definition |
|----------|------------|
| *name* | Optional. Name of the policy set to validate. If a name is not provided then the command will validate the policy set being created or modified in the current session. |

### Example

The first example validates the policy set in the current session. The second example validates the specified *myPolicySet* policy set.

```
wls:/wls-domain/serverConfig>validatePolicySet()
wls:/wls-domain/serverConfig>validatePolicySet('myPolicySet')
```

## validateWSMPolicySet

Command Category: Policy Set Management

Use with WLST: Online

### Description

Within a session, validates an existing policy set. If a policy set name is provided, the specified policy set is validated. If no policy set name is specified, the policy set in the current session is validated.

If the policy set does not exist, if a name is not provided and the session is not active, or if the repository does not contain a suitable policy set, an error message is displayed.

### Syntax

```
validateWSMPolicySet([name=None])
```

| Argument | Definition |
|----------|------------|
| *name* | Optional. Name of the policy set to validate. If a name is not provided then the command will validate the policy set being created or modified in the current session. |

### Examples

The first example validates the policy set in the current session. The second example validates the specified *myPolicySet* policy set.

```
wls:/wls-domain/serverConfig> validateWSMPolicySet()
wls:/wls-domain/serverConfig> validateWSMPolicySet('myPolicySet')
```

## OWSM Repository Management Commands

Use the commands listed in Table 3–10 to manage the Oracle Infrastructure Web Services documents stored in the OWSM repository. For additional information about upgrading or migrating documents in an OWSM repository, see "Upgrading the OWSM Repository" in the *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

---

**Note:** The repository management commands listed in Table 3–11 have been deprecated in this release.

To manage the OWSM repository in release 12*c*, it is recommended that you use the new WLST commands listed in Table 3–10. For a complete list of deprecated commands, see "Deprecated Commands for Oracle Infrastructure Web Services" in *Release Notes for Oracle Fusion Middleware Infrastructure*.

---

Additional MDS WLST commands are described in Chapter 4, "Metadata Services (MDS) Custom WLST Commands."

***Table 3–10   Oracle Infrastructure Web Services - WLST Commands for Repository Management***

| Use this command... | To... | Use with WLST... |
|---|---|---|
| exportWSMAppMetadata | Export a set of applications metadata from the repository into a supported ZIP archive.<br><br>**Note**: This command is supported for Oracle Infrastructure and RESTful web services only. This command is not supported for ADF DC web service clients and Java EE web services. | Online |
| exportWSMRepository | Export a set of documents from the repository into a supported ZIP archive. | Online |
| importWSMArchive | Import a set of documents from a supported ZIP archive into the repository. | Online |
| migrateWSMPMRoles | Migrate the custom roles and policies from the `Plan.xml` file to the `wsm-pm.ear` policy store. | Online |
| migrateWSMAttachments | Migrates direct (local) policy attachments that are identical to the external global policy attachments that would otherwise be attached to each policy subject in the current domain. | Online |
| resetWSMRepository | 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 Fusion Middleware software. | Online |
| upgradeWSMRepository | Upgrade the OWSM predefined policies stored in the repository with any new predefined policies that are provided in the latest installation of the Oracle Fusion Middleware software. | Online |

Table 3–11 list the WLST commands for managing the OWSM repository that have been deprecated in this release.

*Table 3–11    Deprecated WLST Commands for Repository Management*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| exportRepository | Export a set of documents from the repository into a supported ZIP archive. If the specified archive already exists, you can choose whether to overwrite the archive or merge the documents into the existing archive. | Online |
| importRepository | Import a set of documents from a supported ZIP archive into the repository. You can provide the location of a file that describes how to map a physical information from the source environment to the target environment. | Online |
| 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 Fusion Middleware software. | Online |
| upgradeWSMPolicyRepository | Upgrade the OWSM predefined policies stored in the repository with any new predefined policies that are provided in the latest installation of the Oracle Fusion Middleware software. | Online |

## exportRepository

> **Note:** This command has been deprecated. It is recommended that you use the exportWSMRepository command, as described in "exportWSMRepository" on page 3-146. The following examples show how to migrate to use the exportWSMRepository command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> exportRepository
> ("/tmp/repo.zip")
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> exportWSMRepository
> ("/tmp/repo.zip")
> ```

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Exports a set of documents from the OWSM repository into a supported ZIP archive. If the specified archive already exists, the following options are presented:

```
The specified archive already exists. Update existing archive?
Enter "yes" to merge documents into existing archive, "no" to overwrite,
or "cancel" to cancel the operation.
```

You can also specify a list of the documents to be exported, or use a search expression to find specific documents in the repository.

Read only documents, such as predefined policies and assertion templates, will not be included in the export.

### Syntax

```
exportRepository(archive,[documents=None],[includeShared='false'])
```

| Argument | Definition |
|---|---|
| *archive* | Name of the archive file. If the specified archive already exists, you can choose whether to overwrite the archive or merge the documents into the existing archive. |
| | During override, the original archive is backed up and a message describes the location of the backup archive. |
| *documents=None* | Optional. The documents to be exported to the archive. If no documents are specified, then all assertion templates, intents, policies, and policy sets will be exported. You can specify a list of the documents to be exported, or use a search expression to find specific documents in the repository. |
| *includeShared='false'* | Optional. Specifies whether the policy references should be expanded during export. |

## Example

The following examples describe repository export sessions. The first example exports all OWSM documents to the `policies.zip` file.

```
wls:/wls-domain/serverConfig>exportRepository("/tmp/policies.zip")
```

This example exports only the `MyPolicySet1`, `MyPolicySet2`, and `MyPolicySet3` policy sets to the `policies.jar` file, and also expands all the policy references output during the export process.

```
wls:/wls-domain/serverConfig>exportRepository("/tmp/policies.jar",
["/policysets/MyPolicySet1","/policysets/MyPolicySet2","/policysets/MyPolicySet3"]
, true)
```

This example exports policy sets using wildcards to the `some_global_with_noreference_2` file.

```
wls:/wls-domain/serverConfig>exportRepository('./export/some_global_with_
noreference_2', ['policysets:global/web_%','policysets:global/web_ref%',
'policysets:global/web_call%'], false)
```

## exportWSMAppMetadata

> **Note:** This command is supported for Oracle Infrastructure and RESTful web services only. This command is not supported for ADF DC web service clients and Java EE web services.

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Exports a set of application metadata from the repository into a supported ZIP archive. If the specified archive already exists, you are presented with a set of options: merge the documents into the existing archive, overwrite the archive, or cancel the operation. By default, all metadata for applications in the current domain is exported to the archive, or you can use a search expression to export specific metadata for applications in the repository.

> **Note:** Read only documents, such as predefined policies and assertion templates, will not be included in the export.

### Syntax

```
exportWSMAppMetadata(archive,[applications=None],[includeShared='false'])
```

| Argument | Description |
|---|---|
| *archive* | Name of the archive file. If the specified archive already exists, you can choose whether to overwrite the archive or merge the documents into the existing archive. During override, the original archive is backed up and a message describes the location of the backup archive. |
| *applications=None* | Optional. The metadata of applications to be exported to the archive. If no application names are specified, then all metadata for applications in the current domain will be exported. You can specify a list of search expressions to find specific application metadata in the repository, using this syntax: /{*PLATFORM_NAME*}/{*DOMAIN_NAME*}/{*APPLICATION_NAME*}. |
| *includeShared='false'* | Optional. Specifies whether the shared documents (those that are specified as policy references within wsm-assembly documents) should be included during export. Because read-only documents can not be exported, only custom or cloned shared policies will be included in the export. |

### Examples

The first example exports the application metadata in the repository into the applications.zip file and saves it in the tmp directory.

The second example exports the metadata of the applications whose names begin with SalesApp and TradeApp into the applications.zip file and saves it in the tmp directory.

The third example exports the metadata of the applications whose names begin with SalesApp and TradeApp into the applications.zip file and saves it in the tmp directory. Additionally, shared resources are included in this export.

```
wls:/wls-domain/serverConfig> exportWSMAppMetadata("/tmp/applications.zip")

wls:/wls-domain/serverConfig>
exportWSMAppMetadata("/tmp/applications.zip",["/WLS/base_
domain/SalesApp%","WLS/base_domain/TradeApp%"])

wls:/wls-domain/serverConfig>
exportWSMAppMetadata("/tmp/applications.zip",["/WLS/base_
domain/SalesApp%","WLS/base_domain/TradeApp%"], true)
```

## exportWSMRepository

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Exports a set of documents from the OWSM repository into a supported ZIP archive. If the specified archive already exists, the following options are presented:

```
The specified archive already exists. Update existing archive?
Enter "yes" to merge documents into existing archive, "no" to overwrite,
or "cancel" to cancel the operation.
```

You can also specify a list of the documents to be exported, or use a search expression to find specific documents in the repository.

> **Note:** Read only documents, such as predefined policies and assertion templates, will not be included in the export.

### Syntax

```
exportWSMRepository(archive,[documents=None],[includeShared='false'])
```

| Argument | Definition |
|---|---|
| *archive* | Name of the archive file. If the specified archive already exists, you can choose whether to overwrite the archive or merge the documents into the existing archive. |
| | During override, the original archive is backed up and a message describes the location of the backup archive. |
| *documents=None* | Optional. The documents to be exported to the archive. If no documents are specified, then only shared documents that include policies and policy sets will be exported. If this argument is specified as an empty string [''], then all shared documents that include policies and policy sets, application metadata and configuration documents will be exported. You can specify a list of documents to be exported, or use a search expression to find specific documents in the repository. |
| *includeShared='false'* | Optional. Specifies whether the shared documents (those that are specified as policy references within policy sets and wsm-assembly documents) should be included during export. Because read-only documents can not be exported, only custom or cloned shared policies will be included in the export. |

### Examples

The following examples describe repository export sessions. The first example exports all OWSM documents to the policies.zip archive.

```
wls:/wls-domain/serverConfig>exportWSMRepository("/tmp/policies.zip")
```

This example exports only the MyPolicySet1, MyPolicySet2, and MyPolicySet3 policy sets to the policies.jar archive, and also expands all the policy references output during the export process.

```
wls:/wls-domain/serverConfig>exportWSMRepository("/tmp/policies.jar",
["/policysets/MyPolicySet1","/policysets/MyPolicySet2","/policysets/MyPolicySet3"]
```

```
, true)
```

This example exports policy sets using wildcards to the `some_global_with_noreference_2` archive.

```
wls:/wls-domain/serverConfig>exportWSMRepository('./export/some_global_with_noreference_2',
['policysets:global/web_%','policysets:global/web_ref%', 'policysets:global/web_call%'], false)
```

## importRepository

> **Note:** This command has been deprecated. It is recommended that you use the `importWSMArchive` command, as described in "importWSMArchive" on page 3-150. The following examples show how to migrate to use the `importWSMArchive` command.
>
> 11*g* Release (for repository documents):
>
> ```
> wls:/jrfServer_domain/serverConfig> importRepository
> ("/tmp/repo.zip")
> ```
>
> 12*c* Release (for repository documents):
>
> ```
> wls:/jrfServer_domain/serverConfig> importWSMArchive
> ("/tmp/repo.zip")
> ```

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Imports a set of documents from a supported ZIP archive into the OWSM repository. You can use the `map` argument to provide the location of a file that describes how to map physical information from the source environment to the target environment. For example, you can use the map file to ensure that the attachment expression in a policy set document is updated to match the target environment, such as `Domain("foo")=Domain("bar")`.

Read only documents, such as predefined policies and assertion templates, will not be included in the import.

### Syntax

```
importRepository(archive,[map=None],[generateMapFile='false'])
```

| Argument | Definition |
|----------|------------|
| *archive* | Path to the archive file that contains the list of documents to be imported. If a document being imported is a duplicate of the current version that already exists in the repository, then it will not be imported and a new version of the document is not created |
| *map=None* | Optional. Location of a sample map file that describes how to map physical information from the source environment to the target environment. You can generate a new map file by setting the `generateMapFile` argument to `true`.<br><br>If you specify a map file without setting the `generateMapFile` argument to `true`, and the file does not exist, the operation fails and an error is displayed. |

| Argument | Definition |
|----------|------------|
| *generateMapFile=false* | Optional. Specify whether to create a sample map file at the location specified by the map argument. No documents are imported when this argument is set to `true`. The default is `false`. |
| | After the map file is created you can edit it using any text editor. The map file contains the document names given in the archive file and their corresponding `attachTo` values. The `attachTo` value can be updated to correspond to the new environment. If a mapping update is not required for a document name, that entry may be either deleted or commented out using the # character. |
| | **Note:** When importing documents into the repository, OWSM validates the `attachTo` values only. If a value is invalid, then the policy set is disabled. Other text in the map file is not validated. |

## Example

The following examples describe repository import sessions.

The first example imports the contents of the `policies.zip` file into the repository.

```
wls:/wls-domain/serverConfig>importRepository("/tmp/policies.zip")
```

This example uses the `generateMapFile` argument to generate a map file.

```
wls:/wls-domain/serverConfig>importRepository("./export/some_global_with_
noreference_2', map="./export/some_global_with_noreference_2_map',
generateMapFile=true)
```

Here is an example of a generated map file:

```
This is an auto generated override file containing the document names given in
the archive file and their corresponding attachTo values. The attachTo value can
be updated according to the new environment details. If there is no update
required for any document name,that entry may be either deleted or commented
using the character ("#")

[Resource Scope Mappings
]
sca_component_add_1=Composite("*Async*")
sca_reference_add_1=Composite("*Basic_SOA_Client*")
sca_reference_no=Server("*")
sca_service_add_1=Composite("*Basic_SOA_service")
web_callback_add_1=Application("*")
web_client_add_1=Module("*")
web_reference_add_1=Domain("*")
web_service_add_1=Domain("*domain*") and Server("*soa*") and Application("*ADF*")
ws_service_no_1=Server("*Admin*")
```

This example illustrates how to import documents using a generated map file: `/some_global_with_noreference_2_map`.

```
wls:/wls-domain/serverConfig>importRepository('../export/export_all', 'export_all_
map')
```

## importWSMArchive

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Imports a set of documents from a supported ZIP archive into the OWSM repository. You can use the map argument to provide the location of a file that describes how to map physical information from the source environment to the target environment. For example, you can use the map file to ensure that the attachment expression in a policy set document is updated to match the target environment, such as Domain("foo")=Domain("bar").

Read only documents, such as predefined policies and assertion templates, will not be included in the import.

### Syntax

```
importWSMArchive(archive,[map=None],[generateMapFile='false'])
```

| Argument | Definition |
| --- | --- |
| *archive* | Name of the archive file. |
| *map=None* | Optional. Location of a sample map file that describes how to map physical information from the source environment to the target environment. You can generate a new map file by setting the generateMapFile argument to true. |
| | If you specify a map file without setting the generateMapFile argument to true, and the file does not exist, the operation fails and an error is displayed. |
| *generateMapFile=false* | Optional. Specify whether to create a sample map file at the location specified by the map argument. No documents are imported when this argument is set to true. The default is false. |
| | After the file is created you can edit it using any text editor. The attachTo values can be updated to correspond to the new environment. If a mapping update is not required for a document name, that entry may be either deleted or commented out using the # character. |
| | **Note:** When importing documents into the repository, OWSM validates the attachTo values only. If a value is invalid, then the policy set is disabled. Other text in the map file is not validated. |

### Examples

The following examples describe repository import sessions.

The first example imports the contents of the policies.zip file into the repository.

```
wls:/wls-domain/serverConfig>importWSMArchive("/tmp/policies.zip")
```

This example uses the generateMapFile argument to generate a map file.

```
wls:/wls-domain/serverConfig>importWSMArchive("./export/some_global_with_
noreference_2', map="./export/some_global_with_noreference_2_map',
generateMapFile=true)
```

Here is an example of a generated map file:

```
This is an auto generated override file containing the document names given in
the archive file and their corresponding attachTo values. The attachTo value can
be updated according to the new environment details. If there is no update
required for any document name,that entry may be either deleted or commented
using the character ("#")

[Resource Scope Mappings
]
sca_component_add_1=Composite("*Async*")
sca_reference_add_1=Composite("*Basic_SOA_Client*")
sca_reference_no=Server("*")
sca_service_add_1=Composite("*Basic_SOA_service")
web_callback_add_1=Application("*")
web_client_add_1=Module("*")
web_reference_add_1=Domain("*")
web_service_add_1=Domain("*domain*") and Server("*soa*") and Application("*ADF*")
ws_service_no_1=Server("*Admin*")
```

This example illustrates how to import documents using a generated map file: `/some_global_with_noreference_2_map`.

`wls:/wls-domain/serverConfig>`**`importWSMArchive('../export/export_all', 'export_all_map')`**

## migrateWSMPMRoles

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Migrates the custom roles and policies from the `Plan.xml` file to the `wsm-pm.ear` policy store. If the `Plan.xml` file is not used to override default security, then this command will not migrate the `wsm-pm.ear` policy store.

### Syntax

```
migrateWSMPMRoles(domain)
```

| Arguments | Description |
|---|---|
| domain | Absolute path to the domain home where the `wsm-pm` application is configured. |

### Example

In the following example, custom roles and policies are migrated from the `Plan.xml` file to the `wsm-pm.ear` policy store that resides in `'/WLS/myDomain`.

```
wls:/wls-domain/serverConfig> migrateWSMPMRoles('/WLS/myDomain')
```

## migrateWSMAttachments

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Migrates direct (local) policy attachments that are identical to the external global policy attachments that would otherwise be attached to each policy subject in the current domain. You can specify whether to force the migration, prompt for confirmation before each migration, or simply list the migrations that would occur. A direct policy attachment is identical if its URI is the same as one provided by a global policy attachment, and if it does not have any scoped configuration overrides.

> **Note:**  A direct attachment with an unscoped override will be migrated but an attachment with a scoped override will not. This is because after running the migrateAttachments() command, the enforcement of the policies on *all* subjects remains the same, even though some policies are globally attached.

Whether forced or prompted, the command lists each direct policy attachment that is migrated. This output will identify the policy subject that was modified, the URI of the identical policy reference, and the name of the global policy attachment document that duplicated the direct attachment.

### Syntax

```
migrateWSMAttachments([mode='prompt'])
```

| Argument | Definition |
|----------|------------|
| *mode* | The action to be taken for each policy attachment that can be migrated. Valid options are: |
|  | ■     force—Automatically migrate all identical policy attachments without prompting. |
|  | ■     preview—List all policy attachments that can be migrated, but does not perform any migration. |
|  | ■     prompt—Request user confirmation before migrating each policy attachment. |
|  | If no mode is specified, this argument defaults to prompt mode. |

### Examples

The following examples describe how to use the repository attachment migration modes.

```
wls:/wls-domain/serverConfig>migrateWSMAttachments()
wls:/wls-domain/serverConfig>migrateWSMAttachments('force')
wls:/wls-domain/serverConfig>migrateWSMAttachments('preview')
wls:/wls-domain/serverConfig>migrateWSMAttachments('prompt')
```

## resetWSMPolicyRepository

> **Note:** This command has been deprecated. It is recommended that
> you use the resetWSMRepository command, as described in
> "resetWSMRepository" on page 3-155. The following examples show
> how to migrate to use the resetWSMRepository command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> resetWSMPolicyRepository()
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> resetWSMRepository()
> ```

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Deletes the existing policies stored in the OWSM repository and refresh it with the
latest set of predefined policies that are provided in the new installation of the Oracle
Fusion Middleware software. You can use the clearStore argument to specify
whether to delete all policies, including custom user policies, from the OWSM
repository before loading the new predefined policies.

### 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)
```

> **Note:** Use integer values 0 (false) or 1 (true) to pass Boolean types
> on wsadmin and ojbst because the Python version used by these
> scripting tools may not support Boolean types.

## resetWSMRepository

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Deletes the existing policies stored in the repository and refresh it with the current set of predefined policies that are provided in the latest installation of the Oracle Fusion Middleware software. You can use the clearStore argument to specify whether to delete all policies, including custom user policies, from the repository before loading the new predefined policies.

### Syntax

```
resetWSMRepository([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. The repository is then recreated with the new set of predefined documents. |
| | ■ false—Only the predefined policies supplied by Oracle are deleted. Custom documents are *not* deleted when this option is used. The repository is then re-created with the new set of predefined documents. The default is false. |

### Examples

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>resetWSMRepository(true)
```

## upgradeWSMPolicyRepository

> **Note:** This command has been deprecated. It is recommended that you use the `upgradeWSMRepository` command, as described in "upgradeWSMRepository" on page 3-157. The following examples show how to migrate to use the `upgradeWSMRepository` command.
>
> 11*g* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> upgradeWSMPolicyRepository()
> ```
>
> 12*c* Release:
>
> ```
> wls:/jrfServer_domain/serverConfig> upgradeWSMRepository()
> ```

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Upgrades the OWSM predefined policies stored in the OWSM repository with any new predefined policies that are provided in the latest installation of the Oracle Fusion Middleware software. If the repository is empty, all of the predefined policies included in the installation are loaded into the repository.

This command does not remove any existing predefined and user-defined custom policies in the repository. If a predefined policy has been modified or discontinued in a subsequent release, one of the following occurs:

- For policies that have been discontinued, a message is displayed listing the discontinued policies. In this case, Oracle recommends that you no longer reference the policies and remove them using Oracle Enterprise Manager.

- For policies that have changed in the subsequent release, a message is displayed listing the changed policies. Oracle recommends that you import the latest version of the policies using Oracle Enterprise Manager.

### Syntax

```
upgradeWSMPolicyRepository()
```

### Example

The following example upgrades the existing installation with policies provided in the latest release:

```
wls:/wls-domain/serverConfig>upgradeWSMPolicyRepository()
```

## upgradeWSMRepository

Command Category: OWSM Repository Management

Use with WLST: Online

### Description

Upgrades the OWSM predefined policies stored in the repository with any new predefined policies that are provided in the latest installation of the Oracle Fusion Middleware software. If the repository is empty, all of the predefined policies included in the installation are loaded into the repository.

This command does not remove any existing predefined and user-defined custom policies in the repository. If a predefined policy has been modified or discontinued in a subsequent release, one of the following occurs:

- For policies that have been discontinued, a message is displayed listing the discontinued policies. In this case, Oracle recommends that you no longer reference the policies and remove them using Oracle Enterprise Manager.

- For policies that have changed in the subsequent release, a message is displayed listing the changed policies. Oracle recommends that you import the latest version of the policies using Oracle Enterprise Manager.

### Syntax

```
upgradeWSMRepository()
```

### Examples

The following example upgrades the existing installation with policies provided in the latest release:

```
wls:/wls-domain/serverConfig>upgradeWSMRepository()
```

## Token Issuer Trust Configuration Commands

Use the WLST commands listed in Table 3–12 to view and define trusted issuers, trusted distinguished name (DN) lists, and token attribute rules for trusted DNs.

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.

For additional information about using these commands, see "Configuring SAML Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

> **Note:** The commands in this section apply to Oracle Infrastructure Web Services only.
>
> To view the help for the WLST commands described in this section, connect to a running instance of the server and enter `help('wsmManage')`.

***Table 3–12    Web Services 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 |
| 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 trusted issuers, associated DNs, and token attribute rules. | Online |
| importWSMTokenIssuerTrustMetadata | Import trusted issuers, associated DNs, and token attribute rules. | Online |
| listWSMTokenIssuerTrustDocuments | List the token issuer trust documents in the repository. | Online |
| revokeWSMTokenIssuerTrust | Remove trusted issuers, associated 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 token issuer with a DN list. | Online |

*Table 3–12   (Cont.)  Web Services Token Issuer Trust Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| setWSMTokenIssuerTrustAttributeFilter | Add, delete, or update token attribute rules for a given token signing certificate DN. | Online |
| setWSMTokenIssuerTrustAttributeMapping | Set the mapping to map value of an attribute for a trusted DN to local user attribute value and the mapped user attribute. | Online |
| setWSMTokenIssuerTrustDisplayName | Set or reset the display name of the Token Issuer Trust document currently selected in the session. | Online |

## createWSMTokenIssuerTrustDocument

> **Note:** This command applies to Oracle Infrastructure web services only.

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

Within a session, creates a new token issuer trust document using the name provided.

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 SAML Trusted Issuers, DN Lists, and Token Attribute Rules Using WLST" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

### Syntax

```
createWSMTokenIssuerTrustDocument(name, displayName)
```

| Arguments | Definition |
|-----------|------------|
| name | Name of the document to be created. An error is thrown if a name is not provided. |
| 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

> **Note:** This command applies to Oracle Infrastructure web services only.

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

Within a session, deletes the list of all the trusted key identifiers matching the type (such as `dns.hok`, `dns.sv`, or `dns.jwt`) for the issuer specified. 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 | Type of issuer to be deleted, such as `dns.hok`, `dns.sv`, or `dns.jwt`. |
| issuer | Name of the issuer whose trusted DN list will 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')
```

## deleteWSMTokenIssuerTrustAttributeRule

> **Note:** This command applies to Oracle Infrastructure web services only.

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

> **Note:** This command applies to Oracle Infrastructure web services only.

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

Deletes the token issuer trust document, specified by the name argument, from the repository. The default token issuer trust document 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

> **Note:** This command applies to Oracle Infrastructure web services only.

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

Displays the list of all the trusted key identifiers matching the type specified, such as dns.hok, dns.sv, or dns.jwt, and the issuer name.

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

displayWSMTokenIssuerTrust(type, issuer=None)

| Arguments | Definition |
|-----------|------------|
| type | Type of the trusted key identifiers list to be displayed for the issuer. For example, dns.hok, dns.sv, or dns.jwt. |
| issuer | Optional. Name of the trusted issuer for which the trusted key identifiers list is to be displayed. If you do not specify an issuer name, all of the trusted issuers for the given type are listed. |

### Examples

In the following example, the DN lists for the www.example.com trusted issuer are displayed:

```
wls:/wls-domain/serverConfig> displayWSMTokenIssuerTrust('dns.sv',
'www.example.com')
```

## exportWSMTokenIssuerTrustMetadata

> **Note:** This command applies to Oracle Infrastructure web services only.

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

Export the trust configuration (issuers, DNs, and token attribute rules) for all trusted issuers. The configuration will be exported to an XML file identified by the specified location. The configuration for the issuers specified in the exclude list will not be exported. If no argument is passed, trust configuration for all trusted issuers will be exported.

### Syntax

exportWSMTokenIssuerTrustMetadata(trustFile,excludeIssuers=None)

| Arguments | Definition |
|-----------|------------|
| trustFile | Location of the file where the exported metadata will be stored. |
| excludeIssuers | Optional. The list of issuers for which trust metadata should not be exported. |

### Examples

The following examples show the exportWSMTokenIssuerTrustMetadata command.

```
exportWSMTokenIssuerTrustMetadata(trustFile='/tmp/trustData.xml',
  excludeIssuers=['www.example.com','www.myissuer.com'])
```

```
exportWSMTokenIssuerTrustMetadata('/tmp/trustData.xml',['www.example.com'])
```

```
exportWSMTokenIssuerTrustMetadata(trustFile='/tmp/trustData.xml')
```

## importWSMTokenIssuerTrustMetadata

> **Note:** This command applies to Oracle Infrastructure web services only.

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

Import the trust configuration (issuers, DNs, and token attribute rules) for all trusted issuers. The configuration will be imported from the specified XML file.

### Syntax

```
importWSMTokenIssuerTrustMetadata(trustFile)
```

| Argument | Definition |
|----------|------------|
| trustFile | Location of the file from where the configuration will be imported. |

### Examples

The following examples show the importWSMTokenIssuerTrustMetadata command.

```
importWSMTokenIssuerTrustMetadata(trustFile='/tmp/trustData.xml')
```

```
importWSMTokenIssuerTrustMetadata('/tmp/trustData.xml')
```

## listWSMTokenIssuerTrustDocuments

> **Note:** This command applies to Oracle Infrastructure web services only.

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 deleted:

```
wls:/wls-domain/serverConfig> listWSMTokenIssuerTrustDocuments(detail='true')
```

## revokeWSMTokenIssuerTrust

> **Note:** This command applies to Oracle Infrastructure web services only.

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

Remove trusted issuers, associated DNs, and token attribute rules. The issuers specified in the exclude list will not be removed. If no argument is passed, then all trusted issuers and associated configuration will be removed.

### Syntax

```
revokeWSMTokenIssuerTrust(excludeIssuers=None)
```

| Argument | Definition |
|----------|------------|
| excludeIssuers | Optional list of issuers for which the trust configuration should not be removed. |

### Examples

The following examples show the revokeWSMTokenIssuerTrust command.

```
revokeWSMTokenIssuerTrust(excludeIssuers=['www.example.com','www.issuer.com'])
```

```
revokeWSMTokenIssuerTrust(['www.example.com','www.issuer.com'])
```

```
revokeWSMTokenIssuerTrust()
```

## selectWSMTokenIssuerTrustDocument

> **Note:** This command applies to Oracle Infrastructure web services only.

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

> **Note:** This command applies to Oracle Infrastructure web services only.

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

Specify a trusted token issuer with a DN list. This command behaves as follows:

- If the trusted issuer already exists for the type specified, and you provide a list of DNs or aliases for the `trustedKeys` argument, the previous list is replaced with the new list. If you enter an empty set (`[]`) for the `trustedDNs` argument, then the list of DN values are deleted for the issuer.

- If the trusted issuer does not exist for the type specified and you specify a value for the `trustedKeys` argument, the issuer is created with the associated DN list. If you do not set the `trustedKeys` argument, a new issuer is created with an empty DN list.

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

```
setWSMTokenIssuerTrust(type, issuer, [trustedKeys]=None)
```

| Argument | Definition |
|---|---|
| `type` | The type of the tokens issued by the issuer and how the issuer signing the certificates is identified with trusted keys. The following types are supported: <br><br>- `dns.sv`—The token type from the issuer is SAML SV and the trusted key identifier type is X509 Certificate DN. <br><br>- `dns.hok`—The token type from the issuer is SAML HOK or Bearer, and the trusted key identifier type is X509 Certificate DN. <br><br>- `dns.jwt`—The token type from the issuer is JWT, and the trusted key identifier type is X509 Certificate DN. <br><br>- `dns.alias.sv`—The token type from the issuer is SAML SV and the X509 Certificate alias of the issuer signing certificates in the key store is used for trusted key identifier type. <br><br>- `dns.alias.hok`—The token type from the issuer is SAML HOK or Bearer and the X509 Certificate alias of the issuer signing certificates in the key store is used for trusted key identifier type. |
| `issuer` | The name of the trusted issuer, for example `www.example.com`. |
| `trustedKeys` | Optional. List of trusted key identifiers values to set for the specified issuer. |

## Examples

In the following example, `www.yourcompany.com` is set as a trusted issuer and a DN list is not specified:

```
wls:/wls-domain/serverConfig> setWSMTokenIssuerTrust('dns.sv',
'www,yourcompany.com', [])
```

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.example.com` trusted issuer.

```
wls:/wls-domain/serverConfig> setWSMTokenIssuerTrust('dns.sv', 'www.example.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.example.com` trusted issuer:

```
wls:/wls-domain/serverConfig> setWSMTokenIssuerTrust('dns.sv', 'www.example.com',
[])
```

In the following example, the alias `orakey` is specified as the X509 certificate alias for the SAML SV token type for the `www.example.com` trusted issuer:

```
wls:/wls-domain/serverConfig> setWSMTokenIssuerTrust('dn.alias.sv',
'www.example.com', ['orakey'])
```

setWSMTokenIssuerTrustAttributeFilter

## setWSMTokenIssuerTrustAttributeFilter

---

**Note:** This command applies to Oracle Infrastructure web services only.

---

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

Adds, deletes, or updates token attribute rules for a given token signing certificate DN.

Each rule has two parts: a name ID and an attributes part for user attributes that a DN for a signing certificate can assert. The name ID and the attribute can contain a filter with multiple value patterns.

This command behaves as follows:

- If the attribute specified by the attr-name argument already exists with a list of filter values and you provide a new list of values for the filters argument, the previous list is replaced with the new list. If you enter an empty set ([]) for the filters argument, then the existing list of filter values is deleted.

- If the attribute specified by the attr-name argument does not exist and you specify a list of values for the filters argument, the attribute is created and added to the document with the specified filter values. If you do not provide a value for the filters argument, an error is thrown.

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.

---

**Note:** You must first use the setWSMTokenIssuerTrust command to configure a list of trusted DN names for an issuer.

---

### Syntax

```
setWSMTokenIssuerTrustAttributeFilter(dn, attr-name, filters)
```

| Argument | Definition |
|---|---|
| dn | The DN of the token signing certificate. |
| attr-name | The name of the attribute to assert. The value can be as follows:<br><br>■ 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 "*".<br><br>When name-id is selected for the attr-name 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

> **Note:** This command applies to Oracle Infrastructure web services only.

Command Category: Token Issuer Trust Configuration

Use with WLST: Online

### Description

For any DN in the trusted DN list of a trusted token issuer, this command sets the mapping for the attribute (for example, name-id) as specified by the attrName argument. The user attribute argument is optional, and it indicates the local user attribute it corresponds to. The user mapping attribute is also optional and indicates the user attribute to be used in the system to authenticate the users.

### Syntax

```
setWSMTokenIssuerTrustAttributeMapping(dn,attrName,userAttribute=None,
userMappingAttribute=None)
```

| Arguments | Definition |
|---|---|
| dn | DN as the identifier of the token attribute rule where modifications would be done. |
| attrName | Name of the user attribute for which the mapping will be applied. |
| userAttribute | Optional name of the local user attribute the value of the attribute corresponds to. |
| userMappingAttribute | Optional name of the local user attribute to map to. |

### Examples

The following examples show the setWSMTokenIssuerTrustAttributeMapping command.

```
setWSMTokenIssuerTrustAttributeMapping('CN=weblogic, OU=Orakey, O=Oracle, C=US',
'name-id', 'mail', 'uid')
```

```
setWSMTokenIssuerTrustAttributeMapping('CN=weblogic, OU=Orakey, O=Oracle,
C=US','name-id')
```

## setWSMTokenIssuerTrustDisplayName

> **Note:** This command applies to Oracle Infrastructure web services only.

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")
```

# Secure Conversation Session Management Commands

As described in "WS-SecureConversation Architecture", OWSM maintains the client and server secure conversation session information based on a computed Session ID. OWSM (via an internal session mechanism) computes the Session ID at runtime for each message, and associates one or more requests to a session.

Session management commands provide a way for you to release resources on the server when you know that a given client no longer requires the session.

A session is re-used for all requests coming from the same client. In the event a session has been removed manually, a new session is created. If a session is not released manually, it is released the next time that the server hosting the JVM is restarted.

Use the WLST commands listed in Table 3–13 to administer sessions.

For additional information about using these commands, see "Managing Secure Conversation Sessions" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

**Table 3–13    Secure Conversation Session Management Commands**

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| getWebServiceSessionInfo | Display details about the specified active session. | Online |
| listWebServiceSessionNames | List sessions that are currently active for the Session Manager. | Online |
| listWebServiceSessionNamesForKey | List sessions that are active for the Session Manager for a specified key-value pair. | Online |
| removeWebServiceSession | Remove an active session to clear the sessions in a store. | Online |

## getWebServiceSessionInfo

Command Category: Secure Conversation Session Management

Use with WLST: Online

### Description

Gets the specified Session object.    sessionName is returned by
listWebServiceSessionNames().

The returned session names are appropriate for use as the name parameter in
subsequent calls to getWebServiceSessionInfo(String) and
removeWebServiceSession(String) commands.

All of the WebLogic Server instances within a domain must be running in order for
this command to succeed. The scope of the session is the current Persistence provider.

For additional information about using these commands, see "Managing Secure
Conversation Sessions" in *Securing Web Services and Managing Policies with Oracle Web
Services Manager*.

### Syntax

```
getWebServiceSessionInfo ("sessionName")
```

| Arguments | Definition |
|-----------|------------|
| sessionName | Name of the active session for which information is displayed. sessionName is returned by listWebServiceSessionNames(). |

### Examples

In the following example, information about the session named
215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b is returned.

```
wls:/base_domain/serverConfig>
getWebServiceSessionInfo('215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b')
Name: 215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b
Creation time: Mon Nov 04 17:47:39 PST 2013
Last update time: Mon Nov 04 17:47:42 PST 2013
Expiration time: Mon Nov 04 18:17:41 PST 2013
Key info: [oracle.wsm.security.secconv.util.property.SCT,
0x0000014225F1A1260AE4F30351FD1544DC10ED14201988C8CFEDFDBE8E0E4B09]
```

## listWebServiceSessionNames

Command Category: Secure Conversation Session Management

Use with WLST: Online

### Description

Lists the names of all active sessions visible within the domain for the current Persistence provider. The returned list is a snapshot of the visible session instances and is subject to change.

The returned names are appropriate for use as the name parameter in subsequent calls to `getWebServiceSessionInfo()` and `removeWebServiceSession()` commands.

All of the WebLogic Server instances within a domain must be running in order for this command to succeed. The scope of the session is the current Persistence provider.

For additional information about using these commands, see "Managing Secure Conversation Sessions" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

### Syntax

```
listWebServiceSessionNames()
```

### Examples

In the following example, there is one active session.

```
wls:/base_domain/serverConfig> listWebServiceSessionNames()
215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b
```

## listWebServiceSessionNamesForKey

Command Category: Secure Conversation Session Management

Use with WLST: Online

### Description

Lists the names of all sessions that have the name `keyName` and the value `keyValue`. `keyName` and `keyValue` are returned by `getWebServiceSessionInfo()`.

The returned session names are appropriate for use as the name parameter in subsequent calls to `getWebServiceSessionInfo(String)` and `removeWebServiceSession(String)` commands.

All of the WebLogic Server instances within a domain must be running in order for this command to succeed. The scope of the session is the current Persistence provider.

For additional information about using these commands, see "Managing Secure Conversation Sessions" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

### Syntax

```
listWebServiceSessionNamesForKey ("keyName", "keyValue")
```

| Arguments | Definition |
|-----------|------------|
| keyName | A string that specifies the key name for which to list the session names. `keyName` is returned by `getWebServiceSessionInfo()`. |
| keyValue | A string that specifies the key value for which to list the session names. `keyValue` is returned by `getWebServiceSessionInfo()`. |

### Examples

In the following example, there is one active session for the key name `oracle.wsm.security.secconv.util.property.SCT` that has a value of `0x0000014225F1A1260AE4F30351FD1544DC10ED14201988C8CFEDFDBE8E0E4B09`.

```
wls:/base_domain/serverConfig>
listWebServiceSessionNamesForKey('oracle.wsm.security.secconv.util.property.SCT',
'0x0000014225F1A1260AE4F30351FD1544DC10ED14201988C8CFEDFDBE8E0E4B09')
215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b
```

## removeWebServiceSession

Command Category: Secure Conversation Session Management

Use with WLST: Online

### Description

Remove a Session object by giving its name. `sessionName` is returned by `listWebServiceSessionNames()`.

All of the WebLogic Server instances within a domain must be running in order for this command to succeed. The scope of the session is the current Persistence provider.

For additional information about using these commands, see "Managing Secure Conversation Sessions" in *Securing Web Services and Managing Policies with Oracle Web Services Manager*.

### Syntax

```
removeWebServiceSession ("sessionName")
```

| Arguments | Definition |
|---|---|
| sessionName | Name of the active session to remove. `sessionName` is returned by `listWebServiceSessionNames()`. |

### Examples

In the following example, the session named `215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b` is removed.

```
wls:/base_domain/serverConfig>
removeWebServiceSession('215d0d4a5ebbc3fec662f46adedc5bc74ecbc87b')
```

## JKS Keystore Configuration Commands

Use the WLST commands listed in Table 3–14 to view and manage JKS keystore credentials and certificates.

> **Note:** The commands in this section apply to Oracle Infrastructure Web Services only.
>
> To view the help for the WLST commands described in this section, connect to a running instance of the server and enter `help('wsmManage')`.
>
> You must use the OPSS keystore commands if the keystore is KSS. You can view the relevant commands using following command syntax:
>
> ```
> svc = getOpssService(name='KeyStoreService')
> ```
>
> ```
> svc.help()
> ```

*Table 3–14    JKS Keystore Configuration Commands*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| deleteWSMKeyStoreEntry | Delete a single `KeyStore.TrustedCertificateEntry` entry from the keystore. | Online |
| deleteWSMKeyStoreEntries | Delete all `KeyStore.TrustedCertificateEntry` entries from the keystore except those identified by the aliases in the exclusion list. | Online |
| displayWSMCertificate | Displays the string representing the contents of a user's certificate if the alias specifies a `KeyStore.TrustedCertificateEntry`. Displays the certificates in the chain if the alias points to a certificate chain specified by the `KeyStore.PrivateKeyEntry`. | Online |
| exportWSMCertificate | Export a trusted certificate or a certificate chain associated with a private key, indicated by a specified alias, to a specified location. | Online |
| importWSMCertificate | Import a trusted certificate or a certificate chain associated with a private key, indicated by the specified alias. The Base64 encoded certificate will be imported from the specified location. | Online |
| listWSMKeystoreAliases | List all the aliases in the keystore. | Online |

## deleteWSMKeyStoreEntry

> **Note:** This command applies to Oracle Infrastructure Web services only.

Command Category: JKS Keystore Management

Use with WLST: Online

### Description

Delete a single `KeyStore.TrustedCertificateEntry` entry from the keystore. You cannot delete the `keyStore.PrivateKeyEntry`.

### Syntax

```
deleteWSMKeyStoreEntry(alias)
```

| Arguments | Description |
|-----------|-------------|
| *alias* | Alias of the certificate to be deleted. |

### Examples

In this example, the alias for a key store entry, testalias1, is deleted from the keystore.

```
wls:/base_domain/serverConfig> deleteWSMKeyStoreEntry('testalias')

Starting Operation deleteWSMKeyStoreEntry ...
Certificate for alias "testalias" successfully deleted.
```

## deleteWSMKeyStoreEntries

> **Note:** This command applies to Oracle Infrastructure Web services only.

Command Category: JKS Keystore Management

Use with WLST: Online

### Description

Delete all `KeyStore.TrustedCertificateEntry` entries from the keystore except those identified by the aliases in the exclusion list. If no argument is passed then all the `KeyStore.TrustedCertificateEntry` entries will be deleted.

### Syntax

```
deleteWSMKeyStoreEntries(exclusionList=None)
```

| Arguments | Description |
|---|---|
| *exclusionList* | Optional. List of aliases for the certificate that should not be deleted. |

### Examples

In this example, all key store entries are deleted from the keystore, except for the `testalias` and `testalias2` aliases, which are specified on the exclusion list:

```
wls:/base_domain/serverConfig> deleteWSMKeyStoreEntries(['testalias',
'testalias2'])

Starting Operation deleteWSMKeyStoreEntries ...
Certificate(s) deleted successfully.
```

In this example, all key store entries are deleted from the keystore:

```
wls:/base_domain/serverConfig> deleteWSMKeyStoreEntries()
```

## **displayWSMCertificate**

Displays the string representing the contents of a user's certificate if the alias specifies a `KeyStore.TrustedCertificateEntry`. Displays the certificates in the chain if the alias points to a certificate chain specified by a `KeyStore.PrivateKeyEntry`.

> **Note:** This command applies to Oracle Infrastructure Web services only.

Command Category: JKS Keystore Management

Use with WLST: Online

## **Description**

## **Syntax**

```
displayWSMCertificate(alias)
```

| Arguments | Description |
|-----------|-------------|
| *alias* | Alias of the certificate/certificate chain to be displayed. |

## **Examples**

In this example, the contents of the `orakey` trusted certificate is displayed.

```
wls:/base_domain/serverConfig>displayWSMCertificate('orakey')

Starting Operation displayWSMCertificate ...
[
  Version: V3
  Subject: CN=OWSM QA, OU=Fusion Middleware, O=Oracle, L=Redwood City, ST=CA, C=US
  Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5

  Key:  Sun RSA public key, 1024 bits
  modulus:
101336654071087305620295721341875459581727184852017960998615641847764412775989
046768838406911494435712364431883104460420101263455337490958825568587912620074
497379158835791101805994438262634259467352941329678718608662643461089403600239
418798937444529854556507844518713085827283731161032187719240566731105687269
  public exponent: 65537
  Validity: [From: Tue Apr 07 15:04:45 PDT 2009,
             To: Thu Feb 14 14:04:45 PST 2019]
  Issuer: CN=OWSM QA, OU=Fusion Middleware, O=Oracle, L=Redwood City, ST=CA, C=US
  SerialNumber: [    49dbcdfd]
]
  Algorithm: [SHA1withRSA]
  Signature:
0000: 69 29 71 5D 97 1C 28 07   F1 5E 6A AA 49 A7 F7 31  i)q]..(..^j.I..1
0010: F2 B6 91 91 A1 7E D3 F9   1A C6 58 38 85 00 BA 49  .........X8...I
0020: 21 69 E0 06 8D 9F BF 7B   C4 8C 83 95 69 4A 49 EB  !i..........iJI.
0030: 70 D8 7E A9 75 0D 8C C5   7C 9B 14 AB 93 76 A9 35  p...u........v.5
0040: 56 21 71 77 8D 2A AB 1C   CA 81 E0 15 36 4E 81 0A  V!qw.*......6N..
0050: 55 8F D4 5E 1C D0 BF 12   A3 44 8E 65 18 D9 4C E6  U..^.....D.e..L.
0060: 4C 5E 14 4A 7F DF CD 51   59 43 02 41 67 B0 EA 3E  L^.J...QYC.Ag..>
```

```
0070: 58 F4 38 50 3B 2D A3 81   08 8A 84 4C 4B E0 8B 3E   X.8P;-.....LK..>
```

## exportWSMCertificate

---

**Note:** This command applies to Oracle Infrastructure Web services only.

---

Command Category: JKS Keystore Management

Use with WLST: Online

### Description

Export a trusted certificate or a certificate chain associated with a private key indicated by the specified alias. The certificate will be exported to the specified location.

- If the `type` argument is `Certificate`:
  - If the `alias` is pointing to `KeyStore.TrustedCertificateEntry`, it will return the trusted certificate associated with the entry.
  - If the `alias` is pointing to `KeyStore.PrivateKeyEntry`, it will return the first certificate in the certificate chain.
  - If the `alias` does not point to either `KeyStore.TrustedCertificateEntry` or `KeyStore.PrivateKeyEntry`, it will return an error message.
- If the `type` argument is `PKCS7`:
  - If the `alias` is pointing to a `KeyStore.PrivateKeyEntry`, it will return the certificate chain associated with the entry in PKCS7 format.
  - If the `alias` does not point to `KeyStore.PrivateKeyEntry`, it will return an error message.
- If the `type` argument is set to an invalid value, an error message is returned.

### Syntax

```
exportWSMCertificate(alias, certFile, type)
```

| Arguments | Description |
|-----------|-------------|
| *alias* | Alias of the certificate to be exported. |
| *certFile* | Location of the file where the exported certificate will be stored. |
| *type* | Type of keystore entry to be exported. Valid values are:<br>■ `Certificate` for exporting `KeyStore.TrustedCertificateEntry`.<br>■ `PKCS7` for exporting a certificate chain corresponding to a `keyStoreKeyStore.PrivateKeyEntry` specified by the `alias` in PKCS7 format. |

### Examples

In this example, the trusted certificate `testalias` is identified by type as `Certificate` and is exported to the specified `certificate.cer` file:

```
wls:/base_domain/serverConfig>
exportWSMCertificate('testalias','/tmp/certificate.cer','Certificate')

Starting Operation exportWSMCertificate ...
```

```
Certificate for alias "testalias" successfully exported.
```

In this example, the certificate chain `testalias2` is identified by type as `PKCS7` and is exported to the specified `certificatechain.p7b` file:

```
wls:/base_domain/serverConfig>
exportWSMCertificate('testalias2','/tmp/certificatechain.p7b','PKCS7')
```

## importWSMCertificate

> **Note:** This command applies to Oracle Infrastructure Web services only.

Command Category: JKS Keystore Management

Use with WLST: Online

### Description

Import a trusted certificate or a certificate chain associated with a private key indicated by the specified alias. The Base64 encoded certificate will be imported from the specified location.

### Syntax

```
importWSMCertificate(alias, certFile, type, password=None)
```

| Arguments | Description |
|-----------|-------------|
| *alias* | Alias of the certificate to be imported. |
| *certFile* | Location of the file from which the Base64 encoded certificate will be imported. |
| *type* | Type of keystore entry to be imported. Valid values are:<br><br>■ `Certificate` for importing `KeyStore.TrustedCertificateEntry`.<br><br>■ `PKCS7` for importing a certificate chain corresponding to a `keyStoreKeyStore.PrivateKeyEntry` specified by the `alias` in PKCS7 format. |
| *password* | Optional. Password associated with the private key. |

### Examples

In this example, the trusted certificate `testalias` is identified by type as `Certificate` and is imported from the specified `certificate.cer` file:

```
wls:/base_domain/serverConfig>
importWSMCertificate('testalias','/tmp/certificate.cer','Certificate')

Starting Operation importWSMCertificate ...
Certificate for alias "testalias" successfully imported.
```

In this example, the password-protected certificate chain `testalias` is identified by type as `PKCS7` and is imported from the specified `certificatechain.p7b` file:

```
wls:/base_domain/serverConfig>
importWSMCertificate('testalias','/tmp/certificatechain.p7b','PKCS7',password='pri
vatekeypassword')
```

In this example, the certificate chain `testalias` is identified by type as `PKCS7` and is imported from the specified `certificatechain.p7b` file:

```
wls:/base_domain/serverConfig>
importWSMCertificate('testalias','/tmp/certificatechain.p7b','PKCS7')
```

## listWSMKeystoreAliases

> **Note:** This command applies to Oracle Infrastructure Web services only.

Command Category: JKS Keystore Management

Use with WLST: Online

### Description

List all the aliases in the keystore.

### Syntax

```
listWSMKeystoreAliases()
```

### Examples

In this example, all the aliases in the keystore are listed.

```
wls:/base_domain/serverConfig>listWSMKeystoreAliases()

Starting Operation listWSMKeystoreAliases ...

testalias
orakey
testalias2
```

# 4

# Metadata Services (MDS) Custom WLST Commands

This chapter provides detailed descriptions of WLST commands for Oracle Metadata Services (MDS), including command syntax, arguments and command examples.

For additional details about creating and managing an MDS repository, see the chapter "Managing the Oracle Metadata Repository" in *Administering Oracle Fusion Middleware*. For information about the roles needed to perform each operation, see "Understanding MDS Operations" in *Administering Oracle Fusion Middleware*.

Use the Oracle Metadata Services (MDS) commands in the categories listed in Table 4–1 to manage MDS.

*Table 4–1    MDS Command Categories*

| Command category | Description |
| --- | --- |
| Repository Management Commands | Manage the MDS repository. |
| Application Metadata Management Commands | Manage the application metadata in the MDS repository. |
| Sandbox Metadata Management Commands | Manage the metadata in a sandbox in the MDS repository. |
| Application Label Management Commands | Manage the labels for the application. |
| Application Management Deployment Commands | Manage the application deployment. |
| Multitenancy Management Commands | Manage tenants. |

## Common Name Pattern Format

Many commands contain arguments that use name patterns. For example, the restrictCustTo argument use name patterns. The rules for the name patterns are the same for these arguments.

The pattern can contain the following special characters:

- The percent (%) character, which matches any number of characters.

- The underscore (_) character, which matches exactly one arbitrary character.

- The backslash character ('\'), which can be used to escape the percent, the underscore, and the backslash (itself) characters, so they match only %, _, or \.

For example:

```
restrictCustTo="user[scott]"
restrictCustTo="site[site1],user[scott]"
restrictCustTo="site[site1, %_2],user[scott, m%]"
```

# Repository Management Commands

Use the MDS commands listed in Table 4–2 to manage the MDS repository. In the Use with WLST column, online means the command can only be used when connected to a running Administration Server. Offline means the command can only be used when not connected to a running server. Online or offline means the command can be used in both situations.

*Table 4–2    Repository Management Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| createMetadataPartition | Create a metadata repository partition. | Online |
| deleteMetadataPartition | Delete a metadata repository partition. | Online |
| deregisterMetadataDBRepository | Deregister a database-based MDS repository. | Online |
| registerMetadataDBRepository | Register a database-based MDS repository. | Online |

---

**Note:**   Note the following if you invoke the following WLST commands or comparable MBeans in a script:

- registerMetadataDBRepository
- deregisterMetadataDBRepository

In this release and previous releases, the commands or MBeans, have the following behavior:

1. Starts an Oracle WebLogic Server editing session.
2. Registers or deregisters the repository.
3. Activates the changes.

However, you can start an editing session explicitly. If you do, the automatic activation of the changes are deprecated.

---

## createMetadataPartition

Command Category: Repository Management

Use with WLST: Online

### Description

A metadata repository is used as a common repository for managing metadata of different applications. Many applications use the MDS repository to manage their metadata. Each deployed application uses a logical partition in metadata repository. This logical partition also helps in maintaining the metadata lifecycle. Before deploying a application, you create a partition for it in MDS repository. This command creates a partition with the given name in the specified repository.

### Syntax

```
createMetadataPartition(repository, partition)
```

| Argument | Definition |
|----------|------------|
| repository | The name of the repository where the partition will be created. |
| partition | The name of the partition to create in the repository. |

### Example

The following example creates the metadata partition `partition1` in the repository `mds-myrepos`:

```
wls:/weblogic/serverConfig> createMetadataPartition(repository='mds-myrepos',
                              partition='partition1')
Executing operation: createMetadataPartition
Metadata partition created: partition1
"partition1"
```

## deleteMetadataPartition

Command Category: Repository Management

Use with WLST: Online

### Description

Deletes a metadata partition in the specified repository. When you delete a repository partition, all of the metadata in that partition is lost.

### Syntax

deleteMetadataPartition(repository, partition)

| Argument | Definition |
|----------|------------|
| repository | The name of the repository that contains the partition. |
| partition | The name of the partition to delete in the repository. |

### Example

The following example deletes the metadata partition partition1 from the repository mds-myrepos:

```
wls:/weblogic/serverConfig> deleteMetadataPartition(repository='mds-myrepos',
                            partition='partition1')
Executing operation: deleteMetadataPartition
Metadata partition deleted: partition1
```

## deregisterMetadataDBRepository

Command Category: Repository Management

Use with WLST: Online

### Description

Removes the database metadata repository registration as a System JDBC data source in the domain. After this command completes successfully, applications can no longer use this repository.

### Syntax

```
deregisterMetadataDBRepository(name)
```

| Argument | Definition |
|----------|------------|
| name | The name of the repository to deregister. |

### Example

The following example deregisters the metadata repository mds-myrepos:

```
wls:/weblogic/serverConfig> deregisterMetadataDBRepository('mds-myrepos')
Executing operation: deregisterMetadataDBRepository.
Metadata DB repository "mds-myrepos" was deregistered successfully.
```

## registerMetadataDBRepository

Command Category: Repository Management

Use with WLST: Online

### Description

A database metadata repository should be registered with WebLogic Server instances before the application can use it. This command registers a System JDBC data source with the domain for use as database-based metadata repository.

### Syntax

```
registerMetadataDBRepository(name, dbVendor, host, port, dbName, user, password
[, targetServers])
```

| Argument | Definition |
| --- | --- |
| name | The name of the repository to register. If the name you supply does not begin with mds-, the commands adds the prefix mds-. |
| dbVendor | The database vendor. The acceptable values are ORACLE, MSSQL, IBMDB2, and MYSQL. |
| host | The host name or the IP address of the database. |
| port | The port number used by the database. |
| dbName | The service name of the database. For example, orcl.*hostname*.com |
| user | The database user name. |
| password | The password for the database user. |
| targetServers | Optional. The WebLogic Server instances to which this repository will be registered. If this argument is not specified, then the repository will be registered only to the Administration Server. To specify multiple servers, separate the names with a comma. |
| | Register the repository with all Managed Servers to which the application will be deployed. |

### Example

The following example registers the metadata repository myrepos to two servers, and specifies the database parameters:

```
wls:/weblogic/serverConfig> registerMetadataDBRepository('myrepos','ORACLE',
            'test.oracle.com','1521','mds', 'user1','x','server1, server2')
Executing operation: registerMetadataDBRepository.
Metadata DB repository "mds-myrepos" was registered successfully.
'mds-myrepos'
```

## Application Metadata Management Commands

Use the commands in Table 4–3 to manage application metadata.

**Table 4–3    Application Metadata Commands**

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| deleteMetadata | Deletes the metadata in the application repository. | Online |
| exportMetadata | Exports metadata for an application. | Online |
| importMetadata | Imports metadata for an application. | Online |
| purgeMetadata | Purge metadata. | Online |

## deleteMetadata

Command Category: Application Metadata

Use with WLST: Online

### Description

Deletes the selected documents from the application repository. When this command is run against repositories that support versioning (that is, database-based repositories), delete is logical and marks the tip version (the latest version) of the selected documents as "deleted" in the MDS repository partition.

You may want to delete metadata when the metadata is moved from one repository to another. In such a case, after you have exported the metadata, you can delete the metadata in the original repository.

### Syntax

```
deleteMetadata(application, server, docs [, restrictCustTo] [, excludeAllCust]
 [, excludeBaseDocs] [, excludeExtendedMetadata] [, cancelOnException] [,
applicationVersion] [, tenantName])
```

| Argument | Definition |
|----------|------------|
| application | The name of the application for which the metadata is to be deleted. |
| server | The target server on which this application is deployed. |
| docs | A list of comma-separated, fully qualified document names or document name patterns, or both. The patterns can have the following wildcard characters: * and **. |
| | The asterisk (*) represents all documents under the current namespace. The double asterisk (**) represents all documents under the current namespace and also recursively includes all documents in subnamespaces. |
| | For example, "/oracle/*" will include all documents under "/oracle/" but not include documents under "/oracle/mds/". |
| | As another example, "/oracle/**" will include all documents under "/oracle/" and also under "/oracle/mds/" and any other documents further in the namespace chain. |

| Argument | Definition |
|---|---|
| restrictCustTo | Optional. Valid values are percent (%) or a list of comma-separated customization layer names used to restrict the delete operation to delete only customization documents that match the specified customization layers. Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas. |
| | See "Common Name Pattern Format" for information about the patterns that you can use with this argument. |
| | For example: |
| | `restrictCustTo="user[scott]"`<br>`restrictCustTo="site[site1],user[scott]"`<br>`restrictCustTo="site[site1, %_2],user[scott, m%]"` |
| | If you do not specify this argument, only customization classes declared in the cust-config element of adf-config.xml are deleted. If there is no cust-config element declared in adf-config.xml, all customization classes are deleted. |
| | If you specify percent (%) as the value of this argument, all customizations are deleted, whether or not they are declared in the cust-config element of adf-config.xml. |
| | Use this option to delete all customizations or a subset of declared customizations. You can also use this option to delete customizations from customization classes that are not declared in the cust-config element of adf-config.xml. |
| excludeAllCust | Optional. A Boolean value (true or false) that specifies whether or not to delete all customization documents. |
| | This argument defaults to false. It overrides the restrictCustTo option. |
| excludeBaseDocs | Optional. A Boolean value (true or false) that specifies whether or not to delete base documents. This argument defaults to false. |
| excludeExtendedMetadata | Optional. A Boolean value (true or false) that specifies whether or not to delete the Extended Metadata documents. This argument defaults to false. |
| cancelOnException | Optional. A Boolean value (true or false) that specifies whether or not to abort the delete operation when an exception is encountered. On abort, the delete is rolled back if that is supported by the target store. This argument defaults to true. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

## Examples

The following example deletes metadata files under the package `mypackage` from `mdsApp` deployed in the server `server1`:

```
wls:/weblogic/serverConfig> deleteMetadata(application='mdsapp',
          server='server1', docs='/mypackage/*')
Executing operation: deleteMetadata.
"deleteMetadata" operation completed. Summary of "deleteMetadata" operation is:
List of documents successfully deleted:
/mypackage/jobs.xml
```

```
/mypackage/mo.xml
/mypackage/mdssys/cust/site/site1/jobs.xml.xml
/mypackage/mdssys/cust/site/site1/mo.xml.xml
4 documents successfully deleted.
```

The following example deletes metadata files under the package mypackage from mdsApp deployed in the server server1 and excludes extended metadata and all customizations:

```
wls:/weblogic/serverConfig> deleteMetadata(application='mdsapp',
      server='server1', docs='/mypackage/*', cancelOnException='false',
      excludeExtendedMetadata='true',
      excludeAllCust='true')
Executing operation: deleteMetadata.
"deleteMetadata" operation completed. Summary of "deleteMetadata" operation is:
List of documents successfully deleted:
/mypackage/jobs.xml
/mypackage/mo.xml
2 documents successfully deleted.
```

The following example deletes metadata files belonging to tenant tenant1 under the package mypackage from the application app1 deployed in the server server1:

```
wls:/weblogic/serverConfig> deleteMetadata(application='app1', server='server1',
    docs='/mypackage/**', tenantName='tenant1')
Executing operation: deleteMetadata.
deleteMetadata" operation completed. Summary of "deleteMetadata" operation is:
List of documents successfully deleted:
/mypackage/jobs.xml
/mypackage/mdssys/cust/site/site1/jobs.xml.xml
/mypackage/mdssys/cust/site/site2/mo.xml.xml
/mypackage/mdssys/cust/user/user1/mo.xml.xml
```

## exportMetadata

Command Category: Application Metadata

Use with WLST: Online

### Description

Exports application metadata. Use this command and the importMetadata command to transfer application metadata from one server location (for example, testing) to another server location (for example, production).

This command exports application metadata including customizations. However, by default, only those customizations from customization classes that are defined in the cust-config element of adf.config.xml are exported. To export customizations from customization classes not declared, use the restrictCustTo option.

### Syntax

```
exportMetadata(application, server, toLocation  [, docs]
  [, restrictCustTo] [, excludeCustFor] [, excludeAllCust] [, excludeBaseDocs]
  [, excludeExtendedMetadata] [, excludeSeededDocs]
  [, fromLabel][, toLabel] [, applicationVersion] [, remote] [, tenantName])
```

| Argument | Definition |
| --- | --- |
| application | The name of the application from which the metadata is to be exported. |
| server | The target server on which this application is deployed. |
| toLocation | The target directory or archive file (.jar, .JAR, .zip or .ZIP) to which documents selected from the source partition will be transferred. If you export to a directory, the directory must be a local or network directory or file where the application is physically deployed. If you export to an archive, the archive can be located on a local or network directory or file where the application is physically deployed, or on the system on which you are executing the command. |
|  | If the location does not exist in the file system, a directory will be created except that when the names ends with .jar, .JAR, .zip or .ZIP, an archive file will be created. If the archive file already exists, the exportMetadata operation will overwrite the file. |
|  | This argument can be used as temporary file system for transferring metadata from one server to another. For more information, see "Moving Metadata from a Test System to a Production System" in *Administering Oracle Fusion Middleware*. |

| Argument | Definition |
|----------|------------|
| docs | Optional. A list of comma-separated, fully qualified document names or document name patterns, or both. The patterns can have the following wildcard characters: * and **. |
| | This argument defaults to "/**", which exports all the metadata in the repository. |
| | The asterisk (*) represents all documents under the current namespace. The double asterisk (**) represents all documents under the current namespace and also recursively includes all documents in subnamespaces. |
| | For example, "/oracle/*" will include all documents under "/oracle/" but not include documents under "/oracle/mds/". |
| | "/oracle/**" will include all documents under "/oracle/" and also under "/oracle/mds/" and any other documents further in the namespace chain. |
| restrictCustTo | Optional. Valid values are percent (%) or a list of comma-separated customization layer names used to restrict the export operation to export only customization documents that match the specified customization layers. Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas. |
| | See "Common Name Pattern Format" for information about the patterns that you can use with this argument. |
| | For example: |
| | ```
restrictCustTo="user[scott]"
restrictCustTo="site[site1],user[scott]"
restrictCustTo="site[site1, %_2],user[scott, m%]"
``` |
| | If you do not specify this argument, only customization classes declared in the cust-config element of adf-config.xml are exported. If there is no cust-config element declared in adf-config.xml, all customization classes are exported. |
| | If you specify percent (%) as the value of this argument, all customizations are exported, whether or not they are declared in the cust-config element of adf-config.xml. |
| | Use this option to export all customizations or a subset of declared customizations. You can also use this option to export customizations from customization classes that are not declared in the cust-config element of adf-config.xml. |
| | This argument is ignored if the excludeAllCust argument is also specified. |
| excludeCustFor | Optional. A list of comma-separated customization layer names used to restrict the export operation to exclude customization documents that match the specified customization layers from being exported. |
| | This argument is ignored if the excludeAllCust argument is also specified. |
| excludeAllCust | Optional. A Boolean value (true or false) that specifies whether or not to export all customization documents. This argument defaults to false. This argument overrides the restrictCustTo and excludeCustFor arguments. |
| excludeBaseDocs | Optional. A Boolean value (true or false) that specifies whether or not to export base documents. This argument defaults to false. |

| Argument | Definition |
| --- | --- |
| excludeExtendedMetadata | Optional. A Boolean value (true or false) that specifies whether or not to export the Extended Metadata documents. This argument defaults to false. |
| excludeSeededDocs | Optional. A Boolean value (true or false) that specifies whether all documents or only non-seeded documents are exported. Seeded documents are those documents that are packaged in a MAR.<br><br>To exclude seeded documents, specify true.<br><br>The default is false. |
| fromLabel | Optional. Transfers the documents from the source partition that is associated with this label. |
| toLabel | Optional. Works with the fromLabel argument to transfer the delta between fromLabel to toLabel from the source partition. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |
| remote | Optional. A Boolean value (true or false) that specifies whether the archive file will be written to a location where the application is deployed (false) or to the system on which you are executing the command (true).<br><br>The default is false. |
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

## Examples

The following example exports all metadata files from the application `mdsapp` deployed in the server `server1`.

```
wls:/weblogic/serverConfig> exportMetadata(application='mdsapp',
                server='server1',toLocation='/tmp/myrepos',docs='/**')
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean
as the root.
For more help, use help(domainRuntime)
Executing operation: exportMetadata.
"exportMetadata" operation completed. Summary of "exportMetadata" operation is:
List of documents successfully transferred:
/mypackage/write.xml
/mypackage/write1.xml
/sample1.jspx
```

The following example exports only the customization documents under the layer `user` without any base documents from label `label1` to label `label2`:

```
wls:/weblogic/serverConfig> exportMetadata(application='mdsapp',
                     server='server1',toLocation='/tmp/myrepos',
                     restrictCustTo='user',
                     excludeBaseDocs='true',
                     fromLabel='label1',
                     toLabel='label2',
                     applicationVersion='11.1.1')
List of documents successfully transferred:
/mypackage/mdssys/cust/user/user1/write1.xml.xml
/mypackage/mdssys/cust/user/user2/write2.xml.xml
2 documents successfully transferred.
```

## importMetadata

Command Category: Application Metadata

Use with WLST: Online

### Description

Imports application metadata. Use the exportMetadata command and this command to transfer application metadata from one server location (for example, testing) to another server location (for example, production).

### Syntax

```
importMetadata(application, server, fromLocation  [, docs]
 [, restrictCustTo] [, excludeAllCust] [, excludeBaseDocs]
 [, excludeExtendedMetadata] [, excludeUnmodifiedDocs]
  [, cancelOnException] [, applicationVersion] [, remote] [, tenantName])
```

| Argument | Definition |
|---|---|
| application | The name of the application for which the metadata is to be imported. |
| server | The target server on which this application is deployed. |
| fromLocation | The source directory or archive file from which documents will be selected for transfer. If you exported to a directory, the directory must be a local or network directory or file where the application is physically deployed. If you exported to an archive, the archive can be located on a local or network directory or file where the application is physically deployed, or on the system on which you are executing the command.<br><br>This argument can be used as a temporary file system location for transferring metadata from one server to another. For more information, see "Moving Metadata from a Test System to a Production System" in *Administering Oracle Fusion Middleware* |
| docs | Optional. A list of comma-separated, fully qualified document names or document name patterns, or both. The patterns can have the following wildcard characters: * and **.<br><br>This argument defaults to "/**", which imports all of the documents in the repository.<br><br>The asterisk (*) represents all documents under the current namespace. The double asterisk (**) represents all documents under the current namespace and also recursively includes all documents in subnamespaces.<br><br>For example, "/oracle/*" will include all documents under "/oracle/" but not include documents under "/oracle/mds/".<br><br>"/oracle/**" will include all documents under "/oracle/" and also under "/oracle/mds/" and any other documents further in the namespace chain. |

| Argument | Definition |
|---|---|
| restrictCustTo | Optional. Valid values are percent (%) or a list of comma-separated customization layer names used to restrict the import operation to import only customization documents that match the specified customization layers, including customization classes that are not declared in the cust-config element of adf-config.xml. Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas. |
| | See "Common Name Pattern Format" for information about the patterns that you can use with this argument. |
| | For example: |
| | `restrictCustTo="user[scott]"` `restrictCustTo="site[site1],user[scott]"` `restrictCustTo="site[site1, %_2],user[scott, m%]"` |
| | If you do not specify this argument, only customization classes declared in the cust-config element of adf-config.xml are imported. If there is no cust-config element declared in adf-config.xml, all customization classes are imported. |
| | If you specify percent (%) as the value of this argument, all customizations are imported, whether or not they are declared in the cust-config element of adf-config.xml. |
| | Use this option to import all customizations or a subset of declared customizations. You can also use this option to export customizations from customization classes that are not declared in the cust-config element of adf-config.xml. |
| | This argument is ignored if the excludeAllCust argument is also specified. |
| excludeAllCust | Optional. A Boolean value (true or false) that specifies whether or not to import all customization documents. This argument defaults to false. This argument overrides the restrictCustTo argument. |
| excludeBaseDocs | Optional. A Boolean value (true or false) that specifies whether or not to import base documents. This argument defaults to false. |
| excludeExtendedMetadata | Optional. A Boolean value (true or false) that specifies whether or not to import the Extended Metadata documents. This argument defaults to false. |
| excludeUnmodifiedDocs | Optional. A Boolean value (true or false) that specifies whether only changed documents are imported. |
| | If you specify true, only changed documents are imported. |
| | The default is false. |
| cancelOnException | Optional. A Boolean value (true or false) that specifies whether or not to abort the import operation when an exception is encountered. |
| | The default is true. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |
| remote | Optional. A Boolean value (true or false) that specifies whether the archive file is in a location where the application is deployed (false) or on the system on which you are executing the command (true). |
| | The default is false. |

| Argument | Definition |
|---|---|
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

## Example

The following example imports all metadata available in /tmp/myrepos to the application mdsapp deployed in the server server1:

```
wls:/weblogic/serverConfig> importMetadata(application='mdsapp', server='server1',
                           fromLocation='/tmp/myrepos',docs="/**")
Executing operation: importMetadata.
"importMetadata" operation completed. Summary of "importMetadata" operation is:
List of documents successfully transferred:
/app1/jobs.xml
/app1/mo.xml
2 documents successfully transferred.
```

## purgeMetadata

Command Category: Application Metadata

Use with WLST: Online

### Description

Purges the older (non-tip) versions of unlabeled documents from the application's repository. All unlabeled documents will be purged if they are expired, based on Time-To-Live (the olderThan argument). This command is applicable only for repositories that support versioning, that is, database-based repositories.

### Syntax

```
purgeMetadata(application, server, olderThan [, applicationVersion])
```

| Argument | Definition |
| --- | --- |
| application | The name of the application, used to identify the partition in the repository on which the purge operation will be run. |
| server | The target server on which this application is deployed. |
| olderThan | Document versions that are older than this value (in seconds) will be purged. The maximum value is 2147483647 seconds. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |

### Example

The following example purges the document version history for the application mdsapp deployed in the server server1, if the version is older than 10 seconds:

```
wls:/weblogic/serverConfig> purgeMetadata('mdsapp', 'server1', 10)
Executing operation: purgeMetadata.
Metadata purged: Total number of versions: 10.
Number of versions purged: 0.
```

# Sandbox Metadata Management Commands

Use the commands in Table 4–4 to manage metadata in a sandbox. A sandbox is a temporary location for testing changes before moving them to a production system. Sandboxes are not visible to most users until they are applied.

*Table 4–4    Sandbox Metadata Management Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| destroyMDSSandbox | Destroys an MDS sandbox. | Online |
| exportSandboxMetadata | Exports the metadata from a sandbox. | Online |
| importSandboxMetadata | Imports metadata into a sandbox. | Online |
| listMDSSandboxes | Lists sandboxes. | Online |

## destroyMDSSandbox

Command Category: Sandbox Metadata Management

Use with WLST: Online

### Description

Destroys the sandbox and its contents.

You can only use this command with a database-based MDS repository.

### Syntax

```
destroyMDSSandbox(application, server, sandboxName [, applicationVersion])
```

| Argument | Definition |
| --- | --- |
| application | The name of the application. |
| server | The target server on which this application is deployed. |
| sandboxName | The name of the sandbox to destroy. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |

### Example

The following example destroys the sandbox sandbox1 from the MDS repository for the application myapp:

```
wls:/weblogic/serverConfig>destroyMDSSandbox('myapp', 'server1',
                    'sandbox1')
Executing operation: destroyMDSSandbox.

Sandbox "sandbox1" successfully destroyed.
```

## exportSandboxMetadata

Command Category: Sandbox Metadata Management

Use with WLST: Online

### Description

Exports the changes to the metadata from a sandbox on a test system.

You can only use this command with a database-based MDS repository.

### Syntax

```
exportSandboxMetadata(application, server, toArchive, sandboxName
        [, restrictCustTo] [, applicationVersion] [, remote] [, tenantName])
```

| Argument | Definition |
|---|---|
| application | The name of the application from which the metadata is to be exported. |
| server | The target server on which this application is deployed. |
| toArchive | The target archive file (.jar, .JAR, .zip or .ZIP) to which the sandbox contents will be transferred. The archive can be located on a local or network directory where the application is physically deployed. If you specify the -remote argument, the archive can be located on the system on which you are executing the command. |
| sandboxName | The name of the sandbox to export. |
| restrictCustTo | Optional. Valid values are percent (%) or a list of comma-separated customization layer names used to restrict the export operation to export only customization documents that match the specified customization layers. Each customization layer name can contain, within a pair of brackets, optional customization layer values and value patterns separated by commas. |
| | See "Common Name Pattern Format" for information about the patterns that you can use with this argument. |
| | For example: |
| | `restrictCustTo="user[scott]"` `restrictCustTo="site[site1],user[scott]"` `restrictCustTo="site[site1, %_2],user[scott, m%]"` |
| | If you do not specify this argument or if you specify percent (%) as the value of this argument, all customizations are exported, whether or not they are declared in the cust-config element of adf-config.xml. |
| | Use this option to export all customizations or a subset of declared customizations. You can also use this option to export customizations from customization classes that are not declared in the cust-config element of adf-config.xml. |
| | This argument is ignored if the excludeAllCust argument is also specified. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |

| Argument | Definition |
|---|---|
| remote | Optional. A Boolean value (true or false) that specifies whether the archive file will be written to a location where the application is deployed (false) or to the system on which you are executing the command (true). |
| | The default is false. |
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

### Example

The following example exports sandbox sandbox1 from the MDS repository partition for the application myapp to /tmp/sandbox1.jar:

```
wls:/weblogic/serverConfig>exportSandboxMetadata('myapp', 'server1',
                    '/tmp/sandbox1.jar', 'sandbox1')
```

## importSandboxMetadata

Command Category: Sandbox Metadata Management

Use with WLST: Online

### Description

Imports the contents of a sandbox archive to another sandbox in the MDS repository partition of the specified application. It can also update the contents of a given archive to a sandbox in the MDS repository partition of a given application. All customizations are imported, whether or not they are declared in the cust-config element of adf-config.xml.

You can only use this command with a database-based MDS repository.

### Syntax

```
importSandboxMetadata(application, server, fromArchive  [, forceSBCreation]
                [, useExistingSandbox] [, sandboxName] [, applicationVersion]
                [, remote] [, tenantName])
```

| Argument | Definition |
|---|---|
| application | The name of the application for which the metadata is to be imported. |
| server | The target server on which this application is deployed. |
| fromArchive | The source archive file from which documents will be selected for transfer. The archive can be located on a local or network directory where the application is physically deployed. If you specify the -remote argument, the archive can be located on the system on which you are executing the command. |
| forceSBCreation | Optional. A Boolean value (true or false) that specifies whether the operation will overwrite an existing sandbox with the same name. When the argument is set to true, if the fromArchive argument specifies a sandbox with the same name as one that already exists in the application's partition, the original sandbox is deleted and a new sandbox is created. When the argument is set to false, if a sandbox with the same name exists, an exception is thrown. The default is false. |
| useExistingSandbox | Optional. When set to **true**, the contents of the archive are imported to the sandbox specified with the sandboxName argument. This argument is ignored if there is no value specified for sandboxName. The default is false. |
| sandboxName | Optional. The name of the sandbox to update. This argument is ignored if useExistingSandbox is false. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |
| remote | Optional. A Boolean value (true or false) that specifies whether the archive file is in a location where the application is deployed (false) or on the system on which you are executing the command (true). The default is false. |

| Argument | Definition |
|----------|------------|
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

## Examples

The following example imports the contents of sandbox1.jar and creates a sandbox in the MDS repository partition for the application myapp:

```
wls:/weblogic/serverConfig> importSandboxMetadata(application='myapp', 'server1'',
                               '/tmp/sandbox1.jar')
```

The following example updates the existing sandbox sandbox1 in the MDS repository partition for the application myapp with the contents of sandbox1.jar:

```
wls:/weblogic/serverConfig>importSandboxMetadata('myapp', 'server1',
'/tmp/sandbox1.jar', useExistingSandbox='true', sandboxName='sandbox1')
```

## listMDSSandboxes

Command Category: Sandbox Metadata Management

Use with WLST: Online

### Description

Lists sandboxes matching the specified criteria.

You can only use this command with a database-based MDS repository.

### Syntax

```
listMDSSandboxes(application, server [, sbNamePattern] [, applicationVersion])
```

| Argument | Definition |
|----------|------------|
| application | The name of the application whose sandboxes will be listed. |
| server | The target server on which this application is deployed. |
| sbNamePattern | Optional. A pattern that matches the names of one or more sandboxes. When you do not specify this argument, all sandboxes associated with the application's metadata repository partition are listed. |
| | See "Common Name Pattern Format" for information about the patterns that you can use with this argument. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |

### Example

The following example lists all sandboxes for the application myapp and that begin with the characters FlexField:

```
wls:/weblogic/serverConfig>listMDSSandboxes('myapp', 'server1',
                    'FlexField%')
Executing operation: listMDSSandboxes.

Following Sandboxes match the selection criteria:
FlexfieldAutoSandbox_1347601004722
FlexfieldAutoSandbox_1347653193237
FlexfieldAutoSandbox_1347691996491
```

## Application Label Management Commands

Use the commands in Table 4–5 to manage labels for applications.

***Table 4–5    Application Label Management Commands***

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| createMetadataLabel | Creates a metadata label. | Online |
| deleteMetadataLabel | Deletes a metadata label from the repository partition. | Online |
| listMetadataLabels | Lists metadata labels in the repository partition. | Online |
| promoteMetadataLabel | Promotes the metadata associated with a label to tip. | Online |
| purgeMetadataLabels | Deletes the labels matching the specified criteria. | Online |

## createMetadataLabel

Command Category: Application Label Management

Use with WLST: Online

### Description

Creates a new label for the documents in the application's repository partition. This command is applicable only for repositories that support versioning.

### Syntax

```
createMetadataLabel(application, server, name [, applicationVersion] [,
tenantName])
```

| Argument | Definition |
|---|---|
| application | The name of the application for which a label will be created in the partition configured for this application. |
| server | The target server on which this application is deployed. If the application is deployed to multiple Managed Servers in a cluster, you can use the name of any of the server names. You cannot specify multiple server names. |
| name | The name of the label to create in the repository partition. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

### Example

The following example creates the label label1 for the application mdsapp deployed in the server server1:

```
wls:/weblogic/serverConfig> createMetadataLabel('mdsapp','server1','label1')
Executing operation: createMetadataLabel.
Created metadata label "label1".
```

## deleteMetadataLabel

Command Category: Application Label Management

Use with WLST: Online

### Description

Deletes a label for the documents in the application's repository partition. This command is applicable only for repositories that support versioning.

### Syntax

```
deleteMetadataLabel(application, server, name [, applicationVersion] [,
tenantName])
```

| Argument | Definition |
|---|---|
| application | The name of the application from whose associated partition the label is to be deleted. |
| server | The target server on which this application is deployed. If the application is deployed to multiple Managed Servers in a cluster, you can use the name of any of the server names. You cannot specify multiple server names. |
| name | The name of the label to delete in the repository partition. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

### Example

The following example deletes the metadata label label1 from the application mdsapp deployed in the server server1:

```
wls:/weblogic/serverConfig> deleteMetadataLabel('mdsapp','server1','label1')
Executing operation: deleteMetadataLabel.
Deleted metadata label "label1".
```

## listMetadataLabels

Command Category:

Use with WLST: Online

### Description

Lists all of the metadata labels in the application's repository partition. This command is applicable only for repositories that support versioning.

### Syntax

```
listMetadataLabels(application, server [, applicationVersion] [, tenantName])
```

| Argument | Definition |
| --- | --- |
| application | The name of the application for which all of the labels in the repository partition should be listed. |
| server | The target server on which this application is deployed. If the application is deployed to multiple Managed Servers in a cluster, you can use the name of any of the server names. You cannot specify multiple server names. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

### Example

The following example lists the metadata labels available for the application mdsapp deployed in the server server1:

```
wls:/weblogic/serverConfig> listMetadataLabels('mdsapp', 'server1')
Executing operation: listMetadataLabels.
Database Repository partition contains the following labels:
label2
label3
```

## promoteMetadataLabel

Command Category: Application Label Management

Use with WLST: Online

### Description

Promotes documents associated with a label to the tip version in the repository. This command is useful to achieve rollback capability. This command is applicable only for repositories that support versioning.

### Syntax

```
promoteMetadataLabel(application, server, name [, applicationVersion] [,
tenantName])
```

| Argument | Definition |
|---|---|
| application | The name of the application in whose associated repository the metadata is to be promoted to tip. |
| server | The target server on which this application is deployed. If the application is deployed to multiple Managed Servers in a cluster, you can use the name of any of the server names. You cannot specify multiple server names. |
| name | The name of the label to promote in the repository partition. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

### Example

The following example promotes the metadata label label1 to tip in the application mdsapp deployed in the server server1:

```
wls:/weblogic/serverConfig> promoteMetadataLabel('mdsapp', 'server1','label1')
Executing operation: promoteMetadataLabel.
Promoted metadata label "label1" to tip.
```

## purgeMetadataLabels

Command Category: Application Label Management

Use with WLST: Online

### Description

Purges or lists the metadata labels that match the given pattern or age, but does not delete the metadata documents that were part of the label. You can delete the documents by executing the purgeMetadata command.

### Syntax

```
purgeMetadataLabels(repository, partition [, namePattern] [, olderThanInMin]
 [, infoOnly] [, tenantName])
```

| Argument | Definition |
|----------|------------|
| repository | The name of the MDS repository that contains the partition whose metadata labels will be purged or listed. |
| partition | The name of the partition whose metadata labels will be purged or listed. |
| namePattern | Optional. A pattern that matches the names of labels. If you do not specify this argument, all labels in the partition are purged. |
| | See "Common Name Pattern Format" for information about the patterns that you can use with this argument. |
| olderThanInMin | Optional. The age of the labels, in minutes. The default is 525600 (one year). |
| infoOnly | Optional. Valid values are true or false. If you set it to true, it does not purge the labels, but lists the labels that match the specified pattern. |
| | The default is false. |
| tenantName | A unique name identifying the tenant to use for this operation. This argument is required for a multitenant application and is not applicable for a non-multitenant application. For a non-multitenant application, any specified value will be ignored. |

### Examples

The following example lists the labels that match the specified namePattern, but does not delete them:

```
wls:/weblogic/serverConfig> purgeMetadataLabels(repository='mds-myRepos',
          partition='partition1', namePattern='mylabel*', infoOnly='true' )
```

The following example purges the labels that match the specified namePattern and that are older than a year:

```
wls:/weblogic/serverConfig> purgeMetadataLabels(repository='mds-myRepos',
                  partition='partition1', namePattern='mylabel*')
```

The following example deletes labels that match the specified namePattern and that are older than 30 minutes:

```
wls:/weblogic/serverConfig> purgeMetadataLabels(repository='mds-myRepos',
                   partition='partition1',
                 namePattern='mylabel*', olderThanInMin='30')
```

# Application Management Deployment Commands

Use the commands in Table 4–6 to manage deployment.

*Table 4–6    Application Management Deployment Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| getMDSArchiveConfig | Returns an MDSArchiveConfig object. | Offline |
| importMAR | Imports an MAR. | Online |

## getMDSArchiveConfig

Command Category: Application Management Deployment

Use with WLST: Offline

### Description

Returns a handle to the MDSArchiveConfig object for the specified archive. The returned MDSArchiveConfig object's methods can be used to change application and shared repository configuration in an archive.

The MDSArchiveConfig object provides the following methods:

- **setAppMetadataRepository**—This method sets the connection details for the application metadata repository.

  If the archive's existing adf-config.xml file does not contain any configuration for the application's metadata repository, then you must provide all necessary arguments to define the target repository. To define a database-based repository, provide the repository, partition, type, and jndi arguments. For a file-based repository, provide the path argument instead of jndi.

  If the adf-config.xml file already contains some configuration for the application's metadata repository, you can provide only a subset of arguments that you want to change. You do not need to provide all arguments in such a case. However, if the store type is changed, then the corresponding jndi or path argument is required.

- **setAppSharedMetadataRepository**—This method sets the connection details for the shared repository in the application archive that is mapped to specified namespace.

  If the archive's existing adf-config.xml file does not contain any configuration for a shared metadata repository mapped to the specified namespace, you must provide all required arguments (in this case, repository, partition, type, and jndi or path). For a database-based repository, provide the jndi argument. For a file-based repository, path is a required argument.

  If the adf-config.xml file already contains some configuration for a shared metadata repository mapped to the specified namespace and you want to change some specific arguments, you can provide only a subset of those arguments; all others are not needed.

- **save**—If you specify the toLocation argument, then the changes will be stored in the target archive file and the original file will remain unchanged. Otherwise, the changes will be saved in the original file itself.

### Syntax

```
archiveConfigObject = getMDSArchiveConfig(fromLocation)
```

| Argument | Definition |
| --- | --- |
| fromLocation | The name of the ear file, including its complete path. |

The syntax for setAppMetadataRepository is:

```
archiveConfigObject.setAppMetadataRepository([repository] [, partition]
        [, type] [, jndi] [, path])
```

| Argument | Definition |
| --- | --- |
| repository | Optional. The name of the application's repository. |
| partition | Optional. The name of the partition for the application's metadata. |
| type | Optional. The type of connection, file or database, to the repository. Valid values are 'File' or 'DB' (case insensitive). |
| jndi | Optional. The JNDI location for the database connection. This argument is required if the type is set to DB. This argument is not considered if the type is set to File. |
| path | Optional. The directory for the metadata files. This argument is required if the type is set to File. This argument is not considered if the type is set to DB. |

The syntax for setAppSharedMetadataRepository is:

```
archiveConfigObject.setAppSharedMetadataRepository(namespace [, repository]
          [, partition] [, type] [, jndi] [, path])
```

| Argument | Definition |
| --- | --- |
| namespace | The namespace used for looking up the shared repository to set connection details. |
| repository | Optional. The name of the application's shared repository. |
| partition | Optional. The name of the partition for the application's shared metadata. |
| type | Optional. The type of connection, file or database, to the repository. Valid values are 'File' or 'DB' (case insensitive). |
| jndi | Optional. The JNDI location for the database connection. This argument is required if the type is set to DB. This argument will not be considered if the type is set to File. |
| path | Optional. The location of the file metadata store. This argument is required if the type is set to File. This argument will not be considered if the type is set to DB. |

The syntax for save is:

```
archiveConfigObject.save([toLocation])
```

| Argument | Definition |
| --- | --- |
| toLocation | Optional. The file name, including the absolute path to store the changes. If this option is not provided, the changes are written to the archive represented by this configuration object. |

## Examples

In the following example, if the adf-config.xml file in the archive does not have the application and shared metadata repositories defined, then you should provide the complete connection information.

```
wls:/offline> archive = getMDSArchiveConfig(fromLocation='/tmp/testArchive.ear')

wls:/offline> archive.setAppMetadataRepository(repository='AppRepos1',
        partition='partition1', type='DB', jndi='mds-jndi1')

wls:/offline> archive.setAppSharedMetadataRepository(namespace='/a',
        repository='SharedRepos1', partition='partition2', type='File',
```

```
            path='/temp/dir')
wls:/offline> archive.save()
```

In the following example, if the adf-config.xml file in the archive already has the application and shared metadata repositories defined, all arguments are optional. You can set only the arguments you want to change.

```
wls:/offline> archive = getMDSArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setAppMetadataRepository(partition='MDS-partition2')
wls:/offline> archive.setAppSharedMetadataRepository(namespace='/a',
          repository='SharedRepos2')
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

## importMAR

Command Category: Application Management Deployment

Use with WLST: Online

### Description

Imports the metadata from the MAR that is packaged with the application's EAR file. If the MAR had already been imported into the partition, the command deletes the previous version and imports the new version.

### Syntax

```
importMAR(application, server [, force] [, applicationVersion] )
```

| Argument | Definition |
|---|---|
| application | The name of the application for which the metadata is to be imported. |
| server | The target server on which this application is deployed. |
| force | Optional. A Boolean value (true or false) that specifies whether only changed documents and MARs are imported. |
| | For a database-based repository, if you set this argument to false, only new or changed documents from changed MARs are imported. The command creates a label for each MAR for which documents are imported. The label has the following format: |
| | postDeploy_*application_name_MAR_name_MAR_checksum* |
| | For a file-based repository, if you set this argument to false, only changed MARs are imported. The command does not compare individual documents The command creates a file in the repository for each imported MAR. |
| | The default is true. |
| applicationVersion | Optional. The application version, if multiple versions of the same application are deployed. |

### Example

The following example imports metadata from the MAR to the application mdsapp:

```
wls:/weblogic/serverConfig> importMAR('mdsapp','server1')
Executing operation: importMAR.
"importMAR" operation completed. Summary of "importMAR" operation is:
/app1/jobs.xml
/app1/mo.xml
2 documents successfully transferred.
```

## Multitenancy Management Commands

Use the commands in Table 4–7 to manage tenants.

*Table 4–7    Multitenancy Management Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| deprovisionTenant | Deprovisions a tenant from the metadata store. | Online |
| listTenants | Lists the tenants. | Online |

## **deprovisionTenant**

Deprovisions a tenant from the metadata store. All metadata associated with the tenant will be removed from the store

## Syntax

```
deprovisionTenant(repository, partition, tenantName)
```

| Argument | Definition |
|----------|-----------|
| repository | The name of the repository that contains the tenant. |
| partition | The name of the partition that contains the tenant. |
| tenantName | A unique name identifying the tenant to use for this operation. |

## Example

The following example deprovisions the tenant with tenantName tenant1:

```
wls:/weblogic/serverConfig> deprovisionTenant("mds-myrepos", "part1", "tenant1")
Executing operation: deprovisionTenant.
Tenant "tenant1" has been deprovisioned.
```

## listTenants

Lists all tenants in an MDS Repository partition.

### Syntax

```
listTenants(repository, partition)
```

| Argument | Definition |
|----------|------------|
| repository | The name of the repository that contains the tenants. |
| partition | The name of the partition that contains the tenants. |

### Example

The following example lists all tenants in the specified repository and partition:

```
wls:/weblogic/serverConfig> listTenants("mds-myrepos", "part1")
Executing operation: listTenants.
0  GLOBAL
1  tenant1
2  tenant2
3  tenant3
```

# 5

# Application Development Framework (ADF) Custom WLST Commands

The following sections describe the WLST custom commands and variables for Oracle ADF in detail. Topics include:

- Overview of WLST Command Categories
- ADF-Specific WLST Commands
- Using ADF-Specific WLST Commands with Maven

## Overview of WLST Command Categories

Use the ADF-based URL Connections WLST commands to navigate the hierarchy of
configuration or runtime beans and control the prompt display. Use the
`getADFMArchiveConfig` command to manage the `ADFMArchiveConfig` object.

> **Note:** ADF-specific WLST commands can be used with WLST either
> online, offline, or both. Offline WLST commands are not supported
> from Maven.

# ADF-Specific WLST Commands

Use the commands in Table 5–1 to manage URL-based connections.

***Table 5–1    Browse Commands for WLST Configuration***

| Use this command... | To... | Use with WLST... |
|---|---|---|
| adf_createFileUrlConnection | Create a new ADF File connection. | Online or Offline |
| adf_createHttpUrlConnection | Create a new ADF URL connection. | Online or Offline |
| adf_setURLConnectionAttributes | Set or edit the attributes of a newly created or existing ADF connection. | Online or Offline |
| adf_listUrlConnection | List a new URL connection. | Online or Offline |
| getADFMArchiveConfig | Returns a handle to the `ADFMArchiveConfig` object for the specified archive. | Online or Offline |
| exportJarVersions | Export CSV format of JARs versions from current `ORACLE_HOME` at a specified location. | Offline |
| exportApplicationJarVersions | Export CSV format of runtime JARs versions of a specified application at a specified location. | Online |
| exportApplicationSelectedJarVersions | Export CSV format of JARs versions of selected jars at a specified location in coordination with the `Versions.xml` file. | Online |
| createWebServiceConnection | Create a Web service connection for an ADF application. | Online |
| listWebServiceConnection | List Web service connection for an ADF application. | Online |
| deleteWebServiceConnection | Delete a Web service connection for an ADF application. | Online |
| listUpgradeHandlers | List all upgrade handlers of an application. | Online |
| upgradeADFMetadataApp | Upgrade registered ADF Metadata of an application. | Online |
| upgradeADFMetadataAppHandlers | Upgrade selected registered ADF Metadata of an application. | Online |
| upgradeADFMetadata | Upgrade all registered ADF Metadata of all the applications. | Online |
| upgradeADFMetadataHandlers | Upgrade selected registered ADF Metadata of all the applications. | Online |

## adf_createFileUrlConnection

Use with WLST: Online or Offline

### Description

Use this command to create a new connection based on the
`oracle.adf.model.connection.url.FileURLConnection` connection class.

### Syntax

```
adf_createFileURLConnection(appName, name, URL)
```

| Argument | Definition |
|----------|------------|
| *appName* | Application name for which the connection that will be created. |
| *name* | The name of the new connection. |
| *URL* | The URL associated with this connection. |

### Example

```
adf_createFileURLConnection('myapp','tempDir','/scratch/tmp')
```

## adf_createHttpUrlConnection

Use with WLST: Online or Offline

### Description

Use this command to create a new connection based on the
`oracle.adf.model.connection.url.HttpURLConnection` connection type class.

### Syntax

```
adf.createHttpURLConnection (appName, name, [URL], [authenticationType], [realm],
[user], [password]
```

| Argument | Definition |
|---|---|
| *appName* | Application name for which the connection is to be created. |
| *name* | The name of the new connection. |
| *url* | (Optional) The URL associated with this connection. |
| *authenticationType* | (Optional) The default is basic. |
| *realm* | (Optional) If this connection deals with authentication, then this should be set. The default is basic. |
| *user* | (Optional) |
| *password* | (Optional) |

### Example

```
adf_createHttpURLConnection('myapp','cnn','http://www.cnn.com')
```

## adf_setURLConnectionAttributes

Use with WLST: Online or Offline

### Description

Use this command to set or edit the attributes of a newly created or existing ADF connection.

### Syntax

```
adf_setURLConnectionAttributes(appname, connectionname, attributes)
```

| Argument | Definition |
| --- | --- |
| *appname* | Application name for which the connection that will be created. |
| *connectionname* | The name of the new connection. |
| *attributes* | The array containing attributes to set in key/value pairs. |

### Example

```
adf_setURLConnectionAttributes
('myapp','cnn','ChallengeAuthenticationType:digest',
'AuthenticationRealm:XMLRealm'
```

## adf_listUrlConnection

Use with WLST: Online or Offline

### Description

Use this command to list the connections of the application.

### Syntax

```
adf_listURLConnection(appname)
```

| Argument | Definition |
|----------|------------|
| *appname* | Application name |

### Example

```
adf_listURLConnection ('myapp')
```

## getADFMArchiveConfig

Use with WLST: Online or Offline.

### Description

Returns a handle to the `ADFMArchiveConfig` object for the specified archive. The returned `ADFMArchiveConfig` object's methods can be used to change application configuration in an archive.

The `ADFMArchiveConfig` object provides the following methods:

- `setDatabaseJboSQLBuilder([value])`—Sets the Database `jbo.SQLBuilder` attribute.

- `getDatabaseJboSQLBuilder()`—Returns the current value of the `jbo.SQLBuilder` attribute.

- `setDatabaseJboSQLBuilderClass([value])`—Sets the Database `jbo.SQLBuilderClass` attribute. Value is the full name of the custom builder class.

- `getDatabaseJboSQLBuilderClass()`—Returns the current value of the `jbo.SQLBuilderClass` attribute.

- `setDefaultRowLimit([value])`—Sets the defaults `rowLimit` attribute. Value is a long specifying the row limit (Default -1).

- `getDefaultRowLimit()`—Returns the current value of the `rowLimit` attribute.

- `save([toLocation])`—If you specify the `toLocation`, then the changes will be stored in the target archive file and the original file will remain unchanged. Otherwise, the changes will be saved in the original file itself.

### Syntax

```
archiveConfigObject = ADFMAdmin.getADFMArchiveConfig(fromLocation)
```

| Argument | Definition |
| --- | --- |
| *fromLocation* | The name of the ear file, including its complete path. |

The syntax for `setDatabaseJboSQLBuilder([value])` is:

```
archiveConfigObject.setDatabaseJboSQLBuilder([value])
```

| Argument | Definition |
| --- | --- |
| *value* | The value of the `jbo.SQLBuilder` attribute. Valid values are: 'Oracle' (Default), 'OLite', 'DB2', 'SQL92', 'SQLServer', or 'Custom. If 'Custom' is specified, then the `jbo.SQLBuilderClass` attribute should also be set. |

The syntax for `getDatabaseJboSQLBuilder()` is:

```
archiveConfigObject.getDatabaseJboSQLBuilder()
```

The syntax for `setDatabaseJboSQLBuilderClass([value])` is:

```
archiveConfigObject.setDatabaseJboSQLBuilderClass([value])
```

| Argument | Definition |
|----------|------------|
| *value* | The value of the `jbo.SQLBuilderClass` attribute. |

The syntax for `getDatabaseJboSQLBuilderClass)` is:

`archiveConfigObject.getDatabaseJboSQLBuilderClass()`

The syntax for `setDefaultRowLimit([value])` is:

`archiveConfigObject.setDefaultRowLimit([value])`

| Argument | Definition |
|----------|------------|
| *value* | The value of the `rowLimit` attribute. |

The syntax for `getDefaultRowLimit()` is:

`archiveConfigObject.getDefaultRowLimit([value])`

The syntax for `save([toLocation])` is:

`archiveConfigObject.save([toLocation])`

| Argument | Definition |
|----------|------------|
| *toLocation* | The file name along with the absolute path to store the changes. |

## Example

In the following example. the `jbo.SQLBuilder` attribute is set to 'DB2'.

```
wls:/offline> archive =
              ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDatabaseJboSQLBuilder(value='DB2')
wls:/offline> archive.save()
```

In the following example, the `jbo.SQLBuilder` attribute is removed so that application default is used.

```
wls:/offline> archive =
              ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDatabaseJboSQLBuilder()
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

In the following example, the `jbo.SQLBuilder` attribute is set to 'Custom', and the `jbo.SQLBuilderClass` attribute is set to the class 'com.example.CustomBuilder'.

```
wls:/offline> archive =
              ADFMAdmin.getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDatabaseJboSQLBuilder('Custom')
wls:/offline> archive.setDatabaseJboSQLBuilderClass('com.example.CustomBuilder')
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

In the following example, the `rowLimit` attribute is set to 100.

```
wls:/offline> archive = getADFMArchiveConfig(fromLocation='/tmp/testArchive.ear')
wls:/offline> archive.setDefaultRowLimit(100)
wls:/offline> archive.save(toLocation='/tmp/targetArchive.ear')
```

## exportJarVersions

Use with WLST: Offline.

## Description

Use to export CSV format of jars versions from current ORACLE_HOME at a specified location. Exported jars versions information can be opened in Oracle OpenOffice or MS Excel.

## Syntax

```
exportJarVersions(path)
```

| Argument | Definition |
|----------|------------|
| path | Location to extract jars versions. |

## Example

This example shows how jars versions are exported to /tmp/export-MyApp-Versions.csv. R/W privileges for the CSV file need to be verified.

```
wls:/offline>exportJarVersions('/tmp/export-MyApp-Versions.csv')
```

## exportApplicationJarVersions

Use with WLST: Online.

### Description

Used to export CSV format of runtime jars versions of a specified application at a specified location.

### Syntax

```
exportApplicationJarVersions(applicationName, path)
```

| Argument | Definition |
|---|---|
| applicationName | Application name to export jars versions |
| path | Location to export jars versions. |

### Example

This example shows how MyApp runtime jars versions are exported to /tmp/export-MyApp-Versions.csv. R/W privileges for the CSV file need to be verified.

```
wls:/DefaultDomain/serverConfig>exportApplicationJarVersions('MyApp',
'/tmp/export-MyApp-Versions.csv')
```

## exportApplicationSelectedJarVersions

Use with WLST: Online.

### Description

Used to export CSV format of jars versions of selected jars at a specified location.

### Syntax

```
exportApplicationSelectedJarVersions(applicationName, path, jarsLocation)
```

| Argument | Definition |
|---|---|
| applicationName | Application name to export JARs versions. |
| path | Location to extract jars versions. |
| jarsLocation | Optional list of selected JARs. If not specified, default JARs runtime version list from %WLSDOMAIN%/config/fmwconfig/Versions.xml will be exported. |
| | If the selectedJars property in Versions.xml is empty, version information of adfm.jar, adf-richclient-impl-11.jar, adf-controller.jar, adf-pageflow-impl.jar, adf-share-support.jar and mdsrt.jar will be exported. |

### Example

This example shows how JARs versions are exported to /tmp/export-MyApp-Versions.csv using the selectedJars property of the Versions.xml file. In this case, since the jarsLocation parameter is not specified, the libraries listed in the selectedJars property of the Versions.xml file will be exported. R/W privileges for the CSV file need to be verified.

```
wls:/offline>exportApplicationSelectedJarVersions('MyApp',
'/tmp/export-MyApp-Versions.csv')

Versions.xml
<Diagnostics xmlns="xmlns.oracle.com/adf/diagnostics">
  <Versions xmlns="xmlns.oracle.com/adf/diagnostics/versions"
    exportVersionsOnApplicationStartup="true"
    selectedJars="$ORACLE_HOME$/modules/oracle.adf.model_11.1.1/adfm.jar;
      $ORACLE_HOME$/modules/oracle.adf.view_11.1.1/adf-richclient-impl-11.jar;
      $ORACLE_HOME$/modules/oracle.adf.controller_11.1.1/adf-controller.jar;
      $ORACLE_HOME$/modules/oracle.adf.pageflow_11.1.1/adf-pageflow-impl.jar;
      $ORACLE_HOME$/modules/oracle.adf.share_11.1.1/adf-share-support.jar;
      $ORACLE_HOME$/modules/oracle.mds_11.1.1/mdsrt.jar" />
</Diagnostics>
```

This example shows how JARs versions are exported to /tmp/export-MyApp-Versions.csv using the jarsLocation parameter. In this case, the libraries passed explicitly in the jarsLocation parameter will be exported. R/W privileges for the CSV file need to be verified.

```
wls:/offline>exportApplicationSelectedJarVersions('MyApp',
'/tmp/export-MyApp-Versions.csv',
'$ORACLE_HOME$/modules/oracle.adf.model_11.1.1/adfm.jar;$ORACLE_
```

```
HOME$/modules/oracle.adf.view_11.1.1/adf-richclient-impl-11.jar')
```

## createWebServiceConnection

Use with WLST: Online.

### Description

Used to create a Web Service connection for an ADF application.

Returns a set of service name and port names in the format {serviceName: List of portName}.

For example: {'PolicyReferenceEchoBeanService': array(java.lang.String,['PolicyReferenceEchoBeanPort'])}

### Syntax

```
createWebServiceConnection(appName, wsConnName, wsdlUrlStr, readerProps)
```

| Argument | Definition |
|----------|------------|
| appName | Name of the ADF application for which you want to create a Web service connection. |
| wsConnName | Name of the new Web service connection. |
| wsdlUrlStr | Name of the service WSDL URL string. |
| readerProps | The optional WSDL reader properties.<br><br>For example:<br><br>`'["wsdl.reader.proxy.host=proxy.my.com", "wsdl.reader.proxy.port=80"]'` |

### Example

This example shows how to create a Web service connection for an ADF application myapp, with a Web service connection mywsconn, and a WSDL URL string of http://myserver/myservice?WSDL.

```
createWebServiceConnection('myapp','mywsconn','http://myserver/myservice?WSDL')
```

## listWebServiceConnection

Use with WLST: Online.

### Description

Used to list the Web service connections associated with an ADF application.

### Syntax

```
listWebServiceConnection(appName)
```

| Argument | Definition |
| --- | --- |
| appName | Name of the ADF application for which you want to list its Web service connections. |

### Example

This example shows how list the Web service connections for the application myapp.

```
listWebServiceConnection('myapp')
```

## deleteWebServiceConnection

Use with WLST: Online.

### Description

Used to delete a Web service connection associated with an ADF application.

### Syntax

```
listWebServiceConnection(appName, weConnName)
```

| Argument | Definition |
| --- | --- |
| appName | Name of the ADF application for which you want to delete a Web service connection. |
| wsConnName | Name of the Web service connection you want to delete. |

### Example

This example shows how delete the Web service connection mywsconn from the application myapp.

```
deleteWebServiceConnection('myapp','mywsconn')
```

## listUpgradeHandlers

Use with WLST: Online.

### Description

Used to list all upgrade handlers of an application.

### Syntax

```
listUpgradeHandlers(applicationName)
```

| Argument | Definition |
|---|---|
| applicationName | Application name to list upgrade handlers. |

### Example

In the following example, `MyApp` upgrade handlers are listed.

```
wls:/DefaultDomain/serverConfig>listUpgradeHandlers('MyApp')
```

## upgradeADFMetadataApp

Use with WLST: Online.

### Description

Used to upgrade all registered ADF Metadata of an application.

### Syntax

```
upgradeADFMetadataApp(applicationName,siteCC)
```

| Argument | Definition |
| --- | --- |
| applicationName | Application name to upgrade handlers. |
| siteCC | Site Customization Class name. |

### Example

In the following example, all registered ADF Metadata of the application are upgraded.

```
wls:/DefaultDomain/serverConfig>upgradeADFMetadataApp('MyApp','oracle.apps.fnd.app
lcore.customization.SiteCC')
```

## upgradeADFMetadataAppHandlers

Use with WLST: Online.

### Description

Used to upgrade selected registered ADF Metadata of an application.

### Syntax

```
upgradeADFMetadataAppHandlers(applicationName,sitecc,handlers)
```

| Argument | Definition |
|---|---|
| applicationName | Application name to upgrade handlers. |
| siteCC | Site Customization Class name. |
| handlers | Registered handlers to be updated. |

### Example

In the following example, selected registered ADF Metadata of the application are upgraded.

```
wls:/DefaultDomain/serverConfig>upgradeADFMetadataAppHandlers('MyApp','oracle.apps
.fnd.applcore.customization.SiteCC','http://xmlns.oracle.com/adf/metadataUpgrade/b
c4j/propertiesUpgrade')
```

## upgradeADFMetadata

Use with WLST: Online.

### Description

Used to upgrade all registered ADF Metadata of all the applications.

### Syntax

```
upgradeADFMetadata(siteCC)
```

| Argument | Definition |
|----------|------------|
| siteCC | Site Customization Class name. |

### Example

In the following example, all registered ADF Metadata of all the applications are upgraded.

```
wls:/DefaultDomain/serverConfig>upgradeADFMetadata('oracle.apps.fnd.applcore.custo
mization.SiteCC')
```

## upgradeADFMetadataHandlers

Use with WLST: Online.

### Description

Used to upgrade selected registered ADF Metadata of all the applications.

### Syntax

```
upgradeADFMetadataHandlers(sitecc,handlers)
```

| Argument | Definition |
|----------|------------|
| siteCC | Site Customization Class name. |
| handlers | Registered handlers to be updated. |

### Example

In the following example, selected registered ADF Metadata of all the applications are upgraded.

```
wls:/DefaultDomain/serverConfig>upgradeADFMetadataHandlers('oracle.apps.fnd.applco
re.customization.SiteCC','http://xmlns.oracle.com/adf/metadataUpgrade/bc4j/propert
iesUpgrade')
```

# Using ADF-Specific WLST Commands with Maven

The ADF-specific WLST commands can be used with Maven.

To load a set of custom ADF-specific WLST commands, the `com.oracle.adf` groupID and the `adf-wlst-dependencies` artifactId must be specified in the POM's <build> section for the weblogic-maven-plugin as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.test</groupId>
    <artifactId>wlst-test</artifactId>
    <version>1.0-SNAPSHOT</version>
  <build>
    <plugins>
      <plugin>
        <groupId>com.oracle.weblogic</groupId>
        <artifactId>weblogic-maven-plugin</artifactId>
        <version>12.1.4-0-0</version>
        <executions>
          <execution>
            <phase>compile</phase>
            <goals>
              <goal>wlst-client</goal>
            </goals>
            <configuration>
              <fileName>${project.basedir}/misc/test.py</fileName>
            </configuration>
          </execution>
        </executions>
        <dependencies>
          <dependency>
            <groupId>com.oracle.adf</groupId>
            <artifactId>adf-wlst-dependencies</artifactId>
            <version>12.1.4-0-0</version>
          </dependency>
        </dependencies>
      </plugin>
    </plugins>
  </build>

</project>
```

# 6

# DMS Custom WLST Commands

This chapter provides detailed descriptions of custom WLST commands for the Dynamic Monitoring Service (DMS), which you can use to view performance metrics and to configure Event Tracing. This chapter includes command syntax, arguments and command examples.

Use the DMS commands in the categories in Table 6–1 to view performance metrics and to configure Event Tracing.

*Table 6–1    DMS Command Categories*

| Command category | Description |
| --- | --- |
| DMS Configuration Commands | Set and view the DMS system configuration parameters. |
| DMS Metric Commands | View information about performance metrics. |
| DMS Parameter-Scoped Metrics Rules Commands | Create a metric, specifying a constraint and rules that associate sets of noun-types with sets of parameter name |
| DMS Event Tracing Commands | Configure Event Tracing |

# DMS Configuration Commands

Use the commands in Table 6–2 to configure system properties and to display system properties.

*Table 6–2    DMS Configuration Commands*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| listDMSConfigurationParameters | Display information about one or more DMS configuration parameters. | Online |
| setDMSConfigurationParameter | Set the value of a DMS configuration parameter. | Online |

## listDMSConfigurationParameters

Use online.

### Description

Displays information about one or more DMS system configuration parameters.

### Syntax

```
listDMSConfigurationParameters([name][, server])
```

| Argument | Definition |
|----------|------------|
| name | The name of the parameter. |
| server | The name of the server. |

### Examples

The following example displays information about all DMS system configuration parameters:

**listDMSConfigurationParameters()**

```
Server: AdminServer

    Parameter Config Value Runtime Value
    DMSClockType DEFAULT DEFAULT
    SensorActivationLevel NORMAL NORMAL
    DMSClockUnits MICROSECONDS MICROSECONDS
```

The following example displays information about the DMS system configuration parameter DMSClockUnits:

**listDMSConfigurationParameters(name="DMSClockUnits")**

```
Server: AdminServer

    Parameter Config Value Runtime Value
    DMSClockUnits MICROSECONDS MICROSECONDS
```

## setDMSConfigurationParameter

Use online.

### Description

Sets the value of the specified DMS system configuration parameter. This command replaces the existing DMS System Properties, which are now deprecated.

### Syntax

```
setDMSConfigurationParameter(name, value, server)
```

| Argument | Definition |
|----------|------------|
| name | The name of the parameter. |
| value | The value of the parameter. |
| server | The name of the server. |

The following table lists the supported configuration parameters, together with their corresponding system property. These system properties are now deprecated.

| Configuration Parameter | Values | Default | Deprecated System Property |
|-------------------------|--------|---------|----------------------------|
| DMSClockType | default \| highres | default | oracle.dms.clock |
| DMSClockUnits | milliseconds \| microseconds \| nanoseconds | microseconds | oracle.dms.clock.units |
| SensorActivationLevel | none \| normal \| heavy \| all | normal | oracle.dms.sensors |
| DMSPublisherClass | Any string | null (The configuration default) | oracle.dms.publisher.classes |
| DMSHTTPPort | Any numeric port number | 0 (The configuration default) | oracle.dms.httpd.port.start |

### Examples

The following example shows the DMSClockType set to HIGHRES:

**setDMSConfigParameter(name= 'DMSClockType', value='HIGHRES', server='mymanaged')**

## DMS Metric Commands

Use the commands in Table 6–3 to view information about a specific performance metric, a set of performance metrics, or all performance metrics for a particular server or component.

For additional details about metrics, see the chapter "Monitoring Oracle Fusion Middleware" in *Administering Oracle Fusion Middleware* and the chapter "Using the Oracle Dynamic Monitoring Service" in *Tuning Performance*.

*Table 6–3     DMS Metric Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| displayMetricTableNames | Display the names of the available DMS metric tables. | Online |
| displayMetricTables | Display the content of the DMS metric tables. | Online |
| dumpMetrics | Display available metrics. | Online |
| reloadMetricRules | Reload the metric rules. | Online |

## displayMetricTableNames

Command Category: DMS Metrics

Use with WLST: Online

### Description

Displays the names of the available DMS metric tables. The returned value is a list of metric table names.

### Syntax

```
displayMetricTableNames([servers])
```

| Argument | Definition |
|----------|------------|
| servers | Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic Server instance names and system component names. |
| | To specify one server, use the following syntax: |
| | `servers='servername'` |
| | To specify multiple servers, use one of the following syntax options: |
| | `servers=['servername1', 'servername2', ...]`<br>`servers=('servername1', 'servername2', ...)` |
| | If this argument is not specified, the command returns the list of metric table names for all WebLogic servers and system components. |
| | For system components, such as Oracle HTTP Server, use the following format: |
| | `servers=['component_name], servertype='component_type')` |

### Examples

The following example displays metric table names for all WebLogic servers and system components:

```
displayMetricTableNames()
ADF
ADFc
ADFc_Metadata_Service
ADFc_Region
ADFc_Taskflow
ADFc_Viewport
BAM_common_connectionpool
BAM_common_connectionpool_main
BAM_common_messaging
BAM_common_messaging_consumers
.
.
.
```

The following example displays metric table names for the WebLogic Managed Server wls_server1:

```
displayMetricTableNames(servers='wls_server1')
ADF
JVM
JVM_ClassLoader
```

```
JVM_Compiler
JVM_GC
JVM_Memory
JVM_MemoryPool
JVM_MemorySet
JVM_OS
JVM_Runtime
.
.
.
```

The following example displays metric table names for two WebLogic Managed Servers:

**displayMetricTableNames(servers=['wls_server1', 'bam-server1'])**
```
ADF
ADFc
ADFc_Metadata_Service
ADFc_Region
ADFc_Taskflow
ADFc_Viewport
BAM_common_connectionpool
BAM_common_connectionpool_main
BAM_common_messaging
BAM_common_messaging_consumers
.
.
.
```
The following example displays the metric table names for the Oracle HTTP Server instance ohs_1:

**displayMetricTableNames(servers='ohs_1', servertype='OHS')**

## **displayMetricTables**

Command Category: DMS Metrics

Use with WLST: Online

### **Description**

Displays the content of the DMS metric tables.

The returned value is list of DMS metric tables, with the following information about each table:

- The metric table name.

- The metric table schema information.

- The metric table Rows.

The metric table schema information contains the following:

- The name of the column.

- The type of the column value.

- The unit of the column.

- The description of the column.

### **Syntax**

```
displayMetricTables([metricTable_1] [, metricTable_2], [...] [, servers]
                    [, variables])
```

| Argument | Definition |
|---|---|
| metricTable_n | Optional. Specifies a list of metric tables. By default, this argument displays all available metrics. The metric table name can contain special characters for simple pattern matching. The character '?' matches any single character. The character '*' matches zero or more characters. |
| | You specify the metric table name. You can specify multiple metric table names in a comma-separated list. |
| | These are the same names output by the WLST command displayMetricTableNames. |
| servers | Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic Server instance names and system component names. |
| | To specify one server, use the following syntax: |
| | `servers='servername'` |
| | To specify multiple servers, use one of the following syntax options: |
| | `servers=['servername1', 'servername2', ...]`<br>`servers=('servername1', 'servername2', ...)` |
| | If this argument is not specified, the command returns the list of metric tables for all WebLogic servers and system components. |
| | For system components, such as Oracle HTTP Server, use the following format: |
| | `servers=['component_name], servertype='component_type')` |

| Argument | Definition |
|----------|-----------|
| variables | Optional. Defines the metric aggregation parameters. Valid values are a set of name-value pairs. It uses the following syntax: |
| | `variables={name1:value1, name2:value2, ...}` |
| | The specific name-value pairs depend on the aggregated metric tables. Each aggregated metric table has its specific set of variable names. |

## Examples

The following example displays the data from the JVM and the weblogic.management.runtime.WebAppComponentRuntimeMBean metric tables, and limits it to data retrieved from wls_server1 and wls_server2:

```
displayMetricTables('JVM','weblogic.management.runtime.WebAppComponentRuntimeMBean',
    servers=['wls_server1','wlsserver2'])
.
.
.
---
JVM
---

Host:   host.example.com
Name:   JVM
Parent: /
Process:        wls_server_2:7004
ServerName:     wls_server_2
activeThreadGroups.maxValue:    8.0     groups
activeThreadGroups.minValue:    7.0     groups
activeThreadGroups.value:       8       groups
activeThreads.maxValue: 58.0    threads
activeThreads.minValue: 39.0    threads
activeThreads.value:    57      threads
freeMemory.maxValue:    174577.0        kbytes
freeMemory.minValue:    12983.0 kbytes
freeMemory.value:       98562   kbytes
startTime.value:        1368467917680   msecs
.
.
.
```

The following example displays the aggregated metric tables with the specified metric aggregation parameters:

```
displayMetricTables('j2ee_application:webservices_port_rollup',
     servers=['wls_server1','ls_server1'],
     variables={'host':'hostname', 'servletName':'dms'})
----------------------------------------
j2ee_application:webservices_port_rollup
----------------------------------------

Faults: 0
Requests:       0
Requests.averageTime:   0.0
Requests.totalTime:     0.0
ServerName:     wls_server1
moduleName:     RuntimeConfigService
moduleType:     WEBs
portName:       RuntimeConfigServicePortSAML
```

```
processRequest.active:  0
service.throughput:     0.0
service.time:   0.0
startTime:      1238182359291
webserviceName: RuntimeConfigService

Faults: 0
Requests:       0
Requests.averageTime:   0.0
Requests.totalTime:     0.0
ServerName:     wls_server1
moduleName:     TaskMetadataService
moduleType:     WEBs
portName:       TaskMetadataServicePort
processRequest.active:  0
service.throughput:     0.0
service.time:   0.0
startTime:      1238182358096
webserviceName: TaskMetadataService
.
.
.
```

The following example displays the metric tables which names match the specified
patterns:

**displayMetricTables('J??', 'JVM_*')**

```
.
.
.
---------------
JVM_ThreadStats
---------------

Host:   hostname.com
JVM:    JVM
Name:   threads
Parent: /JVM/MxBeans
Process:        AdminServer:7001
ServerName:     AdminServer
contention.value:       enabled in JVM
daemon.value:   85      threads
deadlock.value: 0       threads
live.value:     89      threads
peak.value:     95      threads
started.value:  836     threads
.
.
.
```

## dumpMetrics

Command Category: DMS Metrics

Use with WLST: Online

### Description

Displays available metrics in the internal format or in XML. The returned value is a text document.

### Syntax

```
dumpMetrics([servers] [, format])
```

| Argument | Definition |
|----------|------------|
| servers | Optional. Specifies the servers from which to retrieve metrics. Valid values are a list of WebLogic Server instance names and system component names. |
| | To specify one server, use the following syntax: |
| | `servers='servername'` |
| | To specify multiple servers, use one of the following syntax options: |
| | `servers=['servername1', 'servername2', ...]`<br>`servers=('servername1', 'servername2', ...)` |
| | If this argument is not specified, the command returns the list of metric tables for all WebLogic servers and system components. |
| | For system components, such as Oracle HTTP Server, use the following format: |
| | `servers=['component_name], servertype='component_type')` |
| format | Optional. Specifies the command output format. Valid values are 'raw' (the default), 'xml', and 'pdml'. For example: |
| | `format='raw'`<br>`format='xml'`<br>`format='pdml'` |
| | DMS raw format is a simple metric display format; it displays one metric per line. |

### Examples

The following example outputs all available metrics, including native WebLogic Server metrics and internal DMS metrics, in the XML format:

```
dumpMetrics(format='xml')
<table name='weblogic_j2eeserver:jvm' keys='ServerName serverName'
 componentId='wls_server1' cacheable='false'>
<row cacheable='false'>
<column name='serverName'><![CDATA[wls_server2]]></column>
<column name='nurserySize.value' type='DOUBLE'>0.0</column>
<column name='jdkVersion.value'><![CDATA[1.6.0_05]]></column>
<column name='jdkVendor.value'><![CDATA[BEA Systems, Inc.]]></column>
<column name='daemonThreads.active' type='LONG'>68</column>
<column name='cpuUsage.percentage' type='DOUBLE'>100.0</column>
<column name='threads.active' type='LONG'>71</column>
<column name='ServerName'><![CDATA[wls_server2]]></column>
<column name='heapUsed.value' type='DOUBLE'>0.0</column>
</row>
```

The following example outputs metrics from Server-0 in the default raw format:

**dumpMetrics(servers='Server-0')**
```
.
.
.
    /JVM/MxBeans/threads/Thread-44 [type=JVM
_Thread]
    ECID.value:        null
    RID.value: null
    blocked.value:     0       msec
    blockedCount.value:        1        times
    cpu.value: 40      msecs
    lockName.value:    null
    lockOwnerID.value: null
    lockOwnerName.value:       null
    name.value:        LDAPConnThread-0 ldap://host:7001
    state.value:       RUNNABLE
    waited.value:      0       msec
    waitedCount.value: 0       times
    /JVM/MxBeans/threads/Thread-45 [type=JVM_Thread]
    ECID.value:        null
    RID.value: null
    blocked.value:     0       msec
.
.
.
```

The following example outputs metrics from wls_server1 and wls_server2 in XML format:

**dumpMetrics(servers=['wls_server1', 'wls_server2'], format='xml')**
```
<table name='oracle_soainfra:high_latency_sync_composites' keys='ServerName
 soainfra_composite soainfra_composite_revision soainfra_domain'
 componentId='wls_server2' cacheable='false'>
</table>
<table name='weblogic_j2eeserver:ejb_transaction' keys='ServerName appName
 ejbModuleName name serverName' componentId='wls_server2' cacheable='false'>
<row cacheable='false'>
<column name='serverName'><![CDATA[wls_server2]]></column>
<column name='name'><![CDATA[MessagingClientParlayX]]></column>
<column name='ejbTransactionCommit.percentage' type='DOUBLE'>0.0</column>
<column name='ejbTransactionRollback.completed' type='LONG'>0</column>
<column name='ejbTransactionTimeout.throughput' type='DOUBLE'>0.0</column>
<column name='ejbTransactionCommit.completed' type='LONG'>0</column>
<column name='ejbTransactionTimeout.completed' type='LONG'>0</column>
<column name='appName'><![CDATA[usermessagingserver]]></column>
<column name='ejbTransactionRollback.throughput' type='DOUBLE'>0.0</column>
<column name='ServerName'><![CDATA[wls_server2]]></column>
<column name='ejbTransactionCommit.throughput' type='DOUBLE'>0.0</column>
<column
name='ejbModuleName'><![CDATA[sdpmessagingclient-ejb-parlayx.jar]]></column>
</row>
.
.
.
```

## reloadMetricRules

Command Category: DMS Metrics

Use with WLST: Online

### Description

Reloads the metric rules. You must run this command after you deploy system components or after you modify metric rules. Generally, Oracle does not recommend that you modify metric rules.

### Syntax

```
reloadMetricRules()
```

### Example

The following example reloads metric rules for all servers running in the domain:

```
reloadMetricRules()
Location changed to domainRuntime tree. This is a read-only tree with DomainMBean
 as the root.
For more help, use help(domainRuntime)
loaded 'server-mds-11.0.xml'
loaded 'server-weblogic_j2ee_application_webservices-11.0.xml'
loaded 'server-weblogic_j2eeserver_adf-11.0.xml'
loaded 'server-weblogic_soa_composite-11.0.xml'
loaded 'server-weblogic_j2eeserver_webservices-11.0.xml'
loaded 'server-oracle_sdpmessaging-11.0.xml'
loaded 'server-weblogic_j2ee_application_webcenter-11.0.xml'
loaded 'server-weblogic_j2eeserver-11.0.xml'
reloaded metric rules for server 'wls_server_1'

.
.
.
```

# DMS Parameter-Scoped Metrics Rules Commands

Parameter-scoped metrics commands enable you to create a metric, specifying a constraint and rules that associate sets of noun-types with sets of parameter names.

Use the commands in Table 6–4 to manage parameter-scoped metrics.

*Table 6–4    DMS Parameter-Scoped Metrics Rules Commands*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| createDMSScopedMetricsParameterConstraint | Create a parameter constraint that can be used in the setParameterScopedMetricsRule command. | Online |
| deleteDMSParameterScopedMetricsRules | Delete the specified parameter-scoped metric rules. | Online |
| dumpParameterScopedMetrics | Display the parameter-scoped metric data for the specified rule id. | Online |
| listDMSContextParameters | List the set of execution context parameters known to DMS. | Online |
| listDMSParameterScopedMetricsRules | Display the current parameter-scoped metrics configuration. | Online |
| resetDMSParameterScopedMetrics | Reset the parameter-scoped metric data associated with the given rule identifiers. | Online |
| sampleDMSContextParameterValues | Collect a sample of the set of values for the named context parameter. | Online |
| setDMSParameterScopedMetricsRule | Create or update a parameter-scoped metric rule. | Online |

## createDMSScopedMetricsParameterConstraint

Command Category: DMS Parameter-Scoped Metrics

Use with WLST: Online

### Description

Creates a constraint that can be used with the setParameterScopedMetricsRule command.

### Syntax

```
createDMSScopedMetricsParameterConstraint(name [, values] [,maxnumofvalues])
```

| Argument | Definition |
|---|---|
| name | The name of the parameter to which the constraint refers. |
| values | Optional. A list of specific values to be applied by the constraint. |
| maxmumofvalues | Optional. The maximum number of values that the constraint will use. If omitted, and relevant, a value of 10 is assumed. This value is ignored if the values option is provided. |

### Example

The following example creates the parameter constraint name URI. It applies the values MyApp/advSearch.jspx and MyApp/basicSearch.jspx.

```
createDMSScopedMetricsParameterConstraint(
  name="URI", values=["MyApp/advSearch.jspx", "MyApp/basicSearch.jspx"])
```

## deleteDMSParameterScopedMetricsRules

Command Category: DMS Parameter-Scoped Metrics

Use with WLST: Online

### Description

Deletes the specified parameter-scoped metric rule and its accumulated data.

### Syntax

```
deleteDMSParameterScopedMetricsRules([server,] ids)
```

| Argument | Definition |
|----------|------------|
| server | Optional. Specifies the server from which to delete the parameter-scoped metrics. Valid values are a list of WebLogic Server instance names and system component names. |
| | To specify the server, use the following syntax: |
| | `server='servername'` |
| | If this argument is not specified, the command returns the list of metric tables for all WebLogic servers and system components. |
| | You must be connected to the Administration Server to use this argument. |
| | For system components, such as Oracle HTTP Server, use the following format: |
| | `servers=['component_name], servertype='component_type')` |
| ids | The list of identifiers of the rules to be deleted. To specify more than one identifier, surround the ids with brackets and separate them with commas. For example: |
| | `ids=["id1", "id2"` |

### Example

The following example deletes the parameter-scoped metric rules for the Managed Server wls_server1 and with the id of rule1:

**deleteDMSParameterScopedMetricsRules(server='wls_server1', rule1)**

## dumpParameterScopedMetrics

Command Category: DMS Parameter-Scoped Metrics

Use with WLST: Online

### Description

Displays the parameter-scoped metric data for the specified rule id.

### Syntax

```
dumpParameterScopedMetrics([server,] ruleid)
```

| Argument | Definition |
|----------|------------|
| server | Optional. Specifies the server for which to dump the parameter-scoped metric data. Valid values are a list of WebLogic Server instance names and system component names.<br><br>To specify the server, use the following syntax:<br><br>`server='servername'`<br><br>If this argument is not specified, the command returns the list of metric tables for all WebLogic servers and system components.<br><br>You must be connected to the Administration Server to use this argument.<br><br>For system components, such as Oracle HTTP Server, use the following format:<br><br>`servers=['component_name], servertype='component_type')` |
| ruleid | The id of the parameter-scoped metric rule for which data is to be displayed |

### Example

The following example displays data for the parameter-scoped metric rule regionRule for the Managed Server wls_server1:

**dumpParameterScopedMetrics(server="wls_server1", ruleid="regionRule")**

## listDMSContextParameters

Command Category: DMS Parameter-Scoped Metrics

Use with WLST: Online

### Description

Lists the set of execution context parameters known to DMS.

### Syntax

```
listDMSContextParameters([server][, parameternames] [, verbose])
```

| Argument | Definition |
| --- | --- |
| server | Optional. Specifies the server for which to display the execution context parameters. Valid values are a list of WebLogic Server instance names and system component names. |
| | To specify the server, use the following syntax: |
| | `server='servername'` |
| | If this argument is not specified, the command returns the list of metric tables for all WebLogic servers and system components. |
| | You must be connected to the Administration Server to use this argument. |
| | For system components, such as Oracle HTTP Server, use the following format: |
| | `servers=['component_name], servertype='component_type')` |
| parameternames | Optional. A list of names of execution context parameters of interest. Any parameter name not found on the server will be ignored (no error will be reported). If this argument is not used, all execution context parameters will be listed. |
| verbose | Optional. If `true`, then for each execution context parameter the output will include, where available, the description of the execution context parameter along with its set of possible values and their descriptions. |

### Example

The following example shows the parameter-scoped metric rules for the Managed Server wls_server1:

```
listDMSContextParameters(server='wls_server1')
Server: ManagedServer1

    Module
    FlowId
    Action
    RCID
```

## listDMSParameterScopedMetricsRules

Command Category: DMS Parameter-Scoped Metrics

Use with WLST: Online

### Description

Displays the current parameter-scoped metric configuration.

### Syntax

```
listDMSParameterScopedMetricsRules([server])
```

| Argument | Definition |
|----------|------------|
| server | Optional. Specifies the server for which to display the parameter-scoped metric configuration. Valid values are a list of WebLogic Server instance names and system component names. |
| | To specify the server, use the following syntax: |
| | `server='servername'` |
| | If this argument is not specified, the command returns the list of metric tables for all WebLogic servers and system components. |
| | You must be connected to the Administration Server to use this argument. |
| | For system components, such as Oracle HTTP Server, use the following format: |
| | `servers=['component_name], servertype='component_type')` |

### Example

The following example shows the parameter-scoped metric rules for the Managed Server wls_server1:

**listDMSParameterScopedMetricsRules(server='wls_server1')**

```
 Rule: ruleA
  Noun types:
     JDBC_Connection
  Context Parameter Constraints:
     Parameter: URI
       Constraining values:
         MyApp/advSearch.jspx
         MyApp/basicSearch.jspx
```

## resetDMSParameterScopedMetrics

Command Category: DMS Parameter-Scoped Metrics

Use with WLST: Online

### Description

Resets the parameter-scoped metric data associated with the given rule identifiers. The operation is not guaranteed to be atomic, that is, metric data will continue to be gathered while the reset operation is in progress.

### Syntax

```
resetDMSParameterScopedMetrics([server,] ids)
```

| Argument | Definition |
|---|---|
| server | Optional. Specifies the server for which to reset the parameter-scoped metrics. Valid values are a list of WebLogic Server instance names and system component names. |
| | To specify the server, use the following syntax: |
| | `server='servername'` |
| | If this argument is not specified, the command returns the list of metric tables for all WebLogic servers and system components. |
| | You must be connected to the Administration Server to use this argument. |
| | For system components, such as Oracle HTTP Server, use the following format: |
| | `servers=['component_name], servertype='component_type')` |
| ids | The list of identifiers of the rules to be reset. To specify more than one identifier, surround the ids with brackets and separate them with commas. For example: |
| | `ids=["id1", "id2"` |

### Example

The following example resets the rule with the id ruleA for the managed server wls_server1:

**`resetDMSParameterScopedMetrics(server="wls_server1", ids="ruleA")`**

## sampleDMSContextParameterValues

Command Category: DMS Parameter-Scoped Metrics

Use with WLST: Online

### Description

Collects a sample of the set of values for the named context parameter.

### Syntax

```
sampleDMSContextParameterValues([parametername] [, naxmuofvaleus] [,action])
```

| Argument | Definition |
|---|---|
| parametername | Optional. The name of the context parameter to be sampled. This argument must be used in combination with the argument action. |
| manmumofvalues | Optional. The maximum number of distinct values to include in the sample. If omitted, a value of 10 is applied. A value of 0 will cause all distinct values to be sampled. Depending on the context parameter, this could mean sampling a set of unbounded size. Only relevant when specifying the action="start". |
| action | Optional. The action to be sampled. This argument must be used in combination with the argument parametername.<br><br>The valid values are:<br><br>■    start: Start sampling values<br><br>■    stop: Stop sampling values and show the values<br><br>■    show: Display the histogram of sampled values. |

### Examples

The following example starts sampling the values for the parameter bespoke.ServiceLevel:

```
sampleDMSContextParameterValues(parametername="bespoke.ServiceLevel",
maxnumofvalues=5, action="start")
Values of the parameter bespoke.ServiceLevel are now being sampled.
 List of parameters currently being sampled:
    bespoke.ServiceLevel has been sampled for 0 seconds.
```

The following example stops the sampling and displays the histogram of the values:

```
sampleDMSContextParameterValues()
  List of parameters currently being sampled:    bespoke.ServiceLevel has been
sampled for 87 seconds.
sampleDMSContextParameterValues(parametername="bespoke.ServiceLevel",
action="stop")
  Histogram of values for parameter bespoke.ServiceLevel.
    Gold 21
    Silver 5
    Bronze 37
  Values of the parameter bespoke.ServiceLevel will no longer be sampled.
```

## setDMSParameterScopedMetricsRule

Command Category: DMS Parameter-Scoped Metrics

Use with WLST: Online

### Description

Creates or updates a parameter-scoped metric rule. Only one rule at a time can manage the collection of metrics for a particular noun type and parameter combination. For example, you cannot have two separate rules to attempt to collect parameter-scoped metrics based on the context parameter URI and the noun type JDBC_Connection.

### Syntax

```
setDMSParameterScopedMetricsRule([server,] id, nountypes [, ctxparamconstraints][,
actparamconstraints] [,replace={true|false}])
```

| Argument | Definition |
|---|---|
| server | Optional. Specifies the server for which to set the parameter-scoped metric rule. Valid values are a list of WebLogic Server instance names and system component names. |
| | To specify the server, use the following syntax: |
| | `server='servername'` |
| | If this argument is not specified, the command returns the list of metric tables for all WebLogic servers and system components. |
| | You must be connected to the Administration Server to use this argument. |
| | For system components, such as Oracle HTTP Server, use the following format: |
| | `servers=['component_name], servertype='component_type')` |
| id | The identifier of the rule. |
| nountypes | The list of noun types to which the rule applies. |
| ctxparamconstraints | Optional. The list of context parameter constraints to be used by this rule. You must specify at least one of this argument or actparamconstraints. |
| | See "createDMSScopedMetricsParameterConstraint". |
| actparamconstraints | Optional. The list of activation parameter constraints to be used by the rule. You must specify at least one of this argument or ctxparamconstraints. |
| | See "createDMSScopedMetricsParameterConstraint". |
| replace | Optional. If `true`, the new rule replaces an existing rule of the same id. If `false` or if left unset, and a rule with the id already exists an error is raised. |

### Example

The following example creates the rule with the id ruleA, the nountype JDBC_ Connection and the context parameter constraint ctxP1:

```
setDMSParameterScopedMetricsRule(id="ruleA", nountypes=["JDBC_Connection"],
  ctxparamconstraints=[ctxP1])
```

# DMS Event Tracing Commands

Use the commands in Table 6–5 to configure Event Tracing. Event Tracing configures live tracing with no restarts. DMS metrics that were updated using Oracle Fusion Middleware products may be traced using the DMS Event Tracing feature.

For information about using DMS Event Tracing, see "DMS Tracing and Events" in *Tuning Performance*.

*Table 6–5    DMS Tracing Commands*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| addDMSEventDestination | Add a new destination to the Event Tracing configuration. | Online |
| addDMSEventFilter | Add a filter to the Event Tracing configuration. | Online |
| addDMSEventRoute | Add the specified event route to the Event Tracing configuration | Online |
| enableDMSEventTrace | Enable an event trace and create a filter with a specified condition and destination and an enabled event-route. | Online |
| listDMSEventConfiguration | Display an overview of the event tracing configuration. | Online |
| listDMSEventDestination | Display the full configuration for a destination or a list of all destinations. | Online |
| listDMSEventFilter | Display the configuration of a filter or a list of all filters. | Online |
| listDMSEventRoutes | Display event routes and their status (enabled or disabled). | Online |
| removeDMSEventDestination | Remove the specified destination. | Online |
| removeDMSEventFilter | Remove the specified filter. | Online |
| removeDMSEventRoute | Remove the specified event route. | Online |
| updateDMSEventDestination | Update configuration of an event destination. | Online |
| updateDMSEventFilter | Update the configuration of an event filter. | Online |
| updateDMSEventRoute | Update the configuration of an event route. | Online |

## addDMSEventDestination

Command Category: DMS Event Tracing

Use with WLST: Online

## Description

Adds a new destination to the Event Tracing configuration. If a destination with the same ID already exists, the command reports this and does not add the destination. You must be connected to the Administration Server to add a destination. If you are not, an error is returned.

## Syntax

```
addDMSEventDestination(id [, name] ,class
             [, props= {'name': 'value'...}] [,server])
```

| Argument | Definition |
|----------|-----------|
| id | The unique identifier for the specified destination. |
| name | Optional. A name for the destination. |
| class | The full class name of the destination. See Table 6–6 for a list of available destination classes. |
| props | Optional. The name/value properties to use for the destination. Some destinations require properties, as described in Table 6–6. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

Table 6–6 shows the built-in destinations, with the full runtime class name.

*Table 6–6    Built-In Destinations*

| Runtime Destination Class Name | Description |
|-------------------------------|-------------|
| oracle.dms.trace2.runtime.LoggerDestination | Uses ODL to send the log messages to a file. See Table 6–7 for the list of properties for this destination. |
| oracle.dms.event.HTTPRequestTrackerDestination | Dumps the set of active HTTP requests, allowing an administrator to get a snapshot of activity. See Table 6–8 for the list of properties for this destination. |
| oracle.dms.jrockit.jfr.JFRDestination | Passes events to the JRockit Flight Recorder so that they can be viewed in the context of other data coming from the JRockit JVM and WLDF using JRockit Mission Control. See Table 6–9 for the list of properties for this destination. |
| oracle.dms.jmx.MetricMBeanFactory | Exposes Nouns as MBeans. This destination has no properties. |

*Table 6–6   (Cont.)  Built-In Destinations*

| Runtime Destination Class Name | Description |
|---|---|
| oracle.dms.util.StackTraceCollatorDestination | Collates the stack traces that are in play whenever the events of interest occur. This is primarily a debugging tool. |
| | The collated data is written out on shutdown, and also when an event being handled has not been reported for a certain period of time (defaults to one minute). |
| | See Table 6–10 for the list of properties for this destination. |

Table 6–7 shows the properties for the oracle.dms.trace2.runtime.LoggerDestination destination.

*Table 6–7    Properties for oracle.dms.trace2.runtime.LoggerDestination*

| Property | Description | Valid Values |
|---|---|---|
| LoggerName | Required. The name of the logger. | A valid logger name. |
| writeDataAsMessageAttributes | Optional. If set to true, the event data is logged as supplemental attributes rather than as a colon separated string in the log message. By logging the event data as supplemental attributes, you can exploit the query features of ODL that use supplemental attributes. | true and false |
| | See "Searching Log Files Using WLST" in *Administering Oracle Fusion Middleware* for an example. | |

Table 6–8 shows the properties for the oracle.dms.event.HTTPRequestTrackerDestination destination.

*Table 6–8    Properties for oracle.dms.event.HTTPRequestTrackerDestination*

| Property | Description | Valid Values |
|---|---|---|
| excludeHeaderNames | Optional. Prevents the destination from reporting the specified HTTP request headers if there is a chance that such headers may contain security sensitive information. | A comma-separated list of header names to exclude. |
| requestThresholdSeconds | Optional. The number of seconds after which a request is considered slow. If the generateIncidentMinutes setting is not defined, an incident will be created immediately on detection of a slow request. | A positive numeric value, in seconds. |

*Table 6–8   (Cont.)  Properties for oracle.dms.event.HTTPRequestTrackerDestination*

| Property | Description | Valid Values |
| --- | --- | --- |
| requestFilters | Optional. The filters for specific URIs, or URI+Clicks, to monitor. Each filter has its own threshold. Each filter is defined as follows:<br><br>*thresholdInSeconds*!!*uri*!!*clickId*<br><br>thresholdInSeconds defines the time beyond which a request is considered slow. uri is the URI to be matched, and can be written as a regular expression. clickId is optional and defines the click Id to be matched. It can be written as a regular expression.<br><br>Each filter can be separated using %%. For example:<br><br>/SimpleWebApp-ViewController-context-root/.!!cb.%%/HCM-App/.!!'pt1:AP1:r3:0:AT1:_ATp:resId1:[0-9]+:cl1<br><br>Any requestThresholdSeconds setting will be ignored if requestFilters is specified. | A comma-separated list of request filters |
| generateIncidentMinutes | Optional. The frequency to check if any slow requests have occurred, before generating an incident. Use this setting in conjunction with the requestThresholdSeconds setting. | A positive numeric value, in seconds. |
| incidentSkipCount | Optional. If slow-request detection is enabled (with the requestThresholdSeconds setting, you can use this setting to prevent the creation of the first *n* incidents that would have been created. This is useful for cold servers where there will be additional overhead processing the first few requests. | A positive numeric value, in seconds. |
| maxRequestsReport | Optional. The maximum number of requests to report when generating a delayed incident. Use this setting in conjunction with the generateIncidentMinutes setting. If not defined, all requests will be reported. | A positive numeric value, in seconds. |
| incidentDumps | Optional. The list of diagnostic dumps to execute on detection of a slow request. If this setting is not defined, the default set of diagnostic rules and dumps will be evaluated when creating incidents. | A comma-separated list of diagnostic dump names. |

*Table 6–8   (Cont.)  Properties for oracle.dms.event.HTTPRequestTrackerDestination*

| Property | Description | Valid Values |
|---|---|---|
| dumpIntervalMinutes | Optional. The interval for executing dumps upon detection of the first slow request. This is controlled by this setting in conjunction with the existing `incidentDumps` setting. This allows you to collect diagnostics more frequently than incident creation. For example, you can create an incident every 30 minutes but have dumps collected every 10 minutes.<br><br>The following explains how it works in more detail:<br><br>1.  On detection of the first slow request, the HTTPRequestTracker destination executes the named dumps and then schedules further dumps at the specified frequency.<br><br>2.  At the next dump frequency interval, if there have been new slow requests the dumps will be executed again. If not, no dumps will be executed at this interval.<br><br>3.  At the end of the review period, as specified by the `generateIncidentMinutes` setting, further dumps will be executed if there have been slow requests since the last dump frequency interval and an incident will be created which will include all of the dump files created.<br><br>The incident readme.txt will detail when each dump was taken, | A positive numeric value, in seconds. |
| enablePerformanceMetrics | Optional. After a URI, or URI+Click combination, has been identified as slow, enables additional diagnostics for subsequent requests (in the review period) that match the same URI+Click. The additional diagnostics provide an overview of where time has been spent (for example, JDBC, MDS, ADF) in the request. The data is included in the slowrequests.txt file. | true or false. The default is false. |

Table 6–9 shows the properties for the oracle.dms.jrockit.jfr.JFRDestination destination.

*Table 6–9     Properties for oracle.dms.jrockit.jfr.JFRDestination*

| Property | Description | Valid Values |
|---|---|---|
| maxRecordingSize | Mandatory. The size of the DMS recording (beyond which the flight recorder will drop old data) | An integer followed by K (kilobytes), M (megabytes) or G (gigabytes) |

Table 6–10 shows the properties for the oracle.dms.util.StackTraceCollatorDestination destination.

*Table 6–10    Properties for oracle.dms.util.StackTraceCollatorDestination*

| Property | Description | Valid Values |
|---|---|---|
| printStream | Optional. A string that identifies to which output the print stream will be written | stderr or stdout. The default is stderr. |
| loggerName | Optional. The name of a logger to which output will be written. The destination will check the logger and if no INFO messages will be recorded, the destination will revert to using stderr. | A valid logger name. |
| clearTracesWhenDumped | Optional. Whether or not the destination should reset the set of known stacks and the count of their occurrences once the current data are written out. If false, the set of stacks and counts will accumulate in memory for the lifetime of the destination. | true or false. The default is false. |
| minDumpIntervalInMinutes | Optional. The minimum period of time between writing out data collated for a particular type of event. | A positive numeric value, in minutes. |
| eventTypesOfInterest | Mandatory. A string describing the event types for which stack traces are to be collated. | A valid event type. For example, SENSOR. |

## Examples

The following example adds a destination with the ID destination1, the name File-system, the class oracle.dms.trace2.runtime.LoggerDestination. Because the LoggerDestination requires the property loggerName, it sets the value to trace2-logger:

```
addDMSEventDestination(id='destination1', name='File-system',
                       class='oracle.dms.trace2.runtime.LoggerDestination',
                       props={'loggerName': 'trace2-logger'})
```

```
Destination "destination1" added.
```

The following example attempts to add a destination with an ID that already exists:

```
addDMSEventDestination(id='destination1', name='File-system',
                       class='oracle.dms.trace2.runtime.LoggerDestination',
                       props={'loggerName': 'trace2-logger'})
```

```
Destination "destination1" already exists. Unable to add this.
```

## addDMSEventFilter

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Adds a filter to the Event Tracing configuration. If a filter with the same ID already exists, the command returns an error and does not add the filter.

You must be connected to the Administration Server to add an event filter. If you are not, an error message is reported.

### Syntax

```
addDMSEventFilter(id [, name] [, etypes,]
                    props= {'prop-name': 'value'...} [, server])
```

| Argument | Definition |
|----------|------------|
| id | The unique identifier for specified filter. |
| name | Optional. The name of the filter. |
| etypes | Optional. A string containing a comma-separated list of event/action pairs. This argument allows you to create a filter with a broader granularity when used with a condition. It also allows you to create a filter with a broader range of metrics. For example, all nouns or all nouns with the action create. |
| props | *prop-name*: The name of the filter property. `<condition>` is the only valid property, and only one condition may be specified.<br><br>`value`: The value of the property of the filter. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

The following shows the syntax for `etypes`:

**\<etypes\>:==**
`<type>:[<action>]`

The following lists the valid etypes:

```
NOUN:CREATE|DELETE|TYPE_CHANGE
SENSOR
EVENT_SENSOR:CREATE|DELETE|METRIC_SELECTION_CHANGED,
PHASE_SENSOR:CREATE|DELETE|METRIC_SELECTION_CHANGED|START|STOP|ABORT
STATE_SENSOR:CREATE|DELETE|METRIC_SELECTION_CHANGED|UPDATE
ROLLUP_SENSOR:CREATE|DELETE|METRIC_SELECTION_CHANGED
EXECUTION_CONTEXT:START|STOP|SUSPEND
HTTP_REQUEST:START|STOP|CONTEXT_CHANGED
```

Note the following:

- SENSOR has no associated actions and is expanded to include all related SENSORS and associated actions.

- A type specified with no associated action, will default to all actions for that type. For example, HTTP_REQUEST would default to

HTTP_REQUEST:START,HTTP_REQUEST:STOP, HTTP_REQUEST:CONTEXT_
CHANGED

The following shows an etype with two event/action pairs, separated by a comma:

```
etypes='NOUN:DELETE, STATE_SENSOR:DELETE'
```

The following shows the syntax for the `<condition>` property of the argument `props`. The arguments are described in the tables following the syntax:

**`<condition>::=`**
`<type>  [<operator> <condition>]`

**`<type>::=`**
`<nountype> | <context>`

**`<nountype>::=`**
`NOUNTYPE <nountype-operator> value`

**`<nountype-operator>::=`**
`"equals" | "starts_with" | "contains" | "not_equals"`

**`<context>::=`**
`CONTEXT <name> <context-operator> [<value>] [IGNORECASE=true|false]`
`[DATATYPE="string|long|double"`
`]`
**`<context-operator>::=`**
`"equals" | "starts_with" | "contains" | "not_equals" | "is_null" | "gt" | "le" |`
`"ge"`

**`<operator>::=`**
` AND |OR`

The following table describes the arguments for <type>:

| Value | Description |
|---|---|
| <nountype> | Each Sensor, with its associated metric, is organized in a hierarchy according to Nouns. A Noun type is a name that reflects the set of metrics being collected. For example, JDBC could be a Noun type. For information about Sensors and Nouns, see "Understanding Common DMS Terms and Concepts" in *Tuning Performance*. |
| <context> | An Execution Context is an association of the Execution Context ID (ECID), Relationship ID (RID), and Maps of Values. This argument allows the data stored in the map of values to be inspected and used by the filter. For example, if the map contains the key "user", you can create a filter that returns requests with "user" equal to "bruce". |

The following table describes the arguments for <nountype>:

| Value | Description |
|---|---|
| NOUNTYPE | A keyword. |

| Value | Description |
|---|---|
| <nountype-operator> | The following are valid operators:<br><br>■ equals: Filters only if the Noun type name equals the value.<br><br>■ starts_with: Filters only if the Noun type name starts with the value.<br><br>■ contains: Filters only if the Noun type name equals the value.<br><br>■ not_equals: Filters only if the Noun type name does not equal the value. |
| value | The name of the Noun type on which to operate. The name can be any object for which you want to measure performance. |

The following table describes <context>

| Value | Description |
|---|---|
| CONTEXT | A keyword. |
| name | The name of the context to filter. |
| value | The name of the context on which to operate. |
| <context-operator> | The following are valid operators:<br><br>■ equals: Filters only if the context name equals the value.<br><br>■ starts_with: Filters only if the context name starts with the value.<br><br>■ contains: Filters only if the context name equals the value.<br><br>■ not_equals: Filters only if the context name does not equal the value.<br><br>■ is_null: Filters only if the context name is null.<br><br>■ lt: Filters only if the context name is less than the value.<br><br>■ gt: Filters only if the context name is greater than the value.<br><br>■ le: Filters only if the context name is less than or equal to the value.<br><br>■ ge: Filters only if the context name is greater than or equal to the value. |
| IGNORECASE | Optional. If specified, the case of a string data type is ignored. The default is that the case of a context is used.<br><br>The IGNORECASE AND DATATYPE are not dependent on their position in the command. |
| DATATYPE | Optional. The valid values are string, long, or double. The default is string.<br><br>The IGNORECASE AND DATATYPE are not dependent on their position in the command. |

## Examples

The following example adds a filter with the name MyFilter, specifying a Noun type and context:

```
addDMSEventFilter(id='mds1', name='MyFilter',
        props={'condition': 'NOUNTYPE equals MDS_Connections AND CONTEXT user
equals bruce IGNORECASE'})

Filter "mds1" added.
```

The following example attempts to add a filter with the same id. The command returns an error:

```
addDMSEventFilter(id='mds1', name='MyFilter',
      props={'condition': 'NOUNTYPE equals MDS_Connections AND CONTEXT user equals
bruce'})
```

```
Unable to add filter "mds1" as a filter with that ID already exists for server
"AdminServer".
```

The following example adds a filter with two event/action pairs:

```
addDMSEventFilter(id='mds2', name='MyFilter',
          etypes='NOUN:CREATE,HTTP_REQUEST:START',
          props={'condition': 'NOUNTYPE equals MDS_Connections
                  AND CONTEXT user equals bruce IGNORECASE=true'})
 Filter "mds2" added.
```

## addDMSEventRoute

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Adds the specified event route to the Event Tracing configuration. If an event route with the same ID already exists, the command returns an error and does not add the event route.

You must be connected to the Administration Server to add an event route. If you are not, an error is returned.

### Syntax

```
addDMSEventRoute([filterid,] destinationid [,enable=true|false] [,server])
```

| Argument | Definition |
|----------|------------|
| filterid | Optional. The unique identifier for the filter. |
| destinationid | The unique identifier for the specific destination. The destination must exist. |
| enable | Optional. Enables the filter. Valid values are true and false. The default is true. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Examples

The following example adds an event route with the filter id of mds1 and the destination id of destination1:

```
addDMSEventRoute(filterid='mds1', destinationid='destination1', enable='false')
Event-route for filter "mds1", destination "destination1" added for server
"AdminServer".
```

The following example attempts to add an event route that already exists:

```
addDMSEventRoute(filterid='mds1', destinationid='destination1', enable='false')
Unable to add event route as a mapping with filter "mds1" and destination
"destination1" already exists for server "AdminServer".
```

## enableDMSEventTrace

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Enables an event trace and creates a filter with a specified condition and destination and an enabled event-route. This is a simple way to start filtering, without having to explicitly create a filter, destination and event-route, but with less configuration options. The specified destination must exist.

You must be connected to the Administration Server to enable a DMS event trace. If you are not, an error is returned.

If you require a more complex configuration, use the addDMSEventDestination, addDMSEventFilter, and addDMSEventRoute.

### Syntax

```
enableDMSEventTrace(destinationid [, etypes] [, condition]  [, server])
```

| Argument | Definition |
|----------|------------|
| destinationid | The unique identifier for the specific destination. Any existing destination is valid. |
| etypes | Optional. A string containing a comma-separated list of event/action pairs. See addDMSEventFilter for a list of available etypes. |
| condition | Optional. A condition on which to filter. See addDMSEventFilter for the syntax for a condition. <br><br> If no condition is specified, all DMS events will be passed |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Example

The following example enables an event trace with a specified condition:

```
enableDMSEventTrace(destinationid='destination1', condition='CONTEXT username
EQUALS Joe AND CONTEXT ip EQUALS 192.168.1.5')

Filter "auto215443800" using Destination "destination1" added, and event-route
enabled for server "AdminServer".
```

## listDMSEventConfiguration

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Displays an overview of the Event Tracing configuration.

### Syntax

```
listDMSEventConfiguration([server])
```

| Argument | Definition |
|----------|------------|
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Example

The following example lists the configuration for the Managed Server to which you are connected:

**listDMSEventConfiguration()**

```
Server: AdminServer

Event routes:
   Filter       :  auto215443800
   Destination  :  destination1
   Enabled      :  true
```

## listDMSEventDestination

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

For a specific destination, display the full configuration. If no destination ID is specified, list the destination ID and name for all the destinations in the Event Tracing configuration.

### Syntax

```
listDMSEventDestination([id] [, server])
```

| Argument | Definition |
|----------|------------|
| id | Optional. The unique identifier for the specific destination. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Examples

The following example displays information about the destinations for the Managed Server to which you are connected:

**listDMSEventDestination()**
```
Server: AdminServer

   Id            : HTTPRequestTrackerDestination
   Name          : HTTP Request Tracker Destination


   Id            : mbeanCreationDestination
   Name          : MBean Creation Destination
```

The following example displays information about the destinations for the Managed Server, wls_server_1:

**listDMSEventDestination(server='wls_server_1')**
```
Server: wls_server_1

   Id            : HTTPRequestTrackerDestination
   Name          : HTTP Request Tracker Destination


   Id            : mbeanCreationDestination
   Name          : MBean Creation Destination
 .
 .
 .
```

The following example displays information about the destination destination1:

**listDMSEventDestination(id='destination1')**
```
Server: AdminServer

   Id            : destination1
```

```
   Name          : File-system
   Class         : oracle.dms.trace2.runtime.LoggerDestination
   Class Info    : Logs incoming events to the logger configured for the
Destination.
   Properties    :
      Name                        Value
      loggerName                  trace2-logger
```

## listDMSEventFilter

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

For a specific filter, displays the full configuration. If you do not specify a filter ID, the command displays the filter ID and name for all the filters in the Event Tracing configuration.

### Syntax

```
listDMSEventFilter([id] [, server])
```

| Argument | Definition |
|---|---|
| id | Optional. The unique identifier for specified filter. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Example

The following example displays the list of all the filters in the Event Tracing configuration:

```
listDMSEventFilter()
   Id                        Name
   auto215443800             auto generated using enableEventTrace
   JFRFilter                 JFRFilter
   traceFilter
   mds2                      MyFilter
   mds1                      MyFilter
```

The following example displays the configuration of the filter mds1:

```
listDMSEventFilter(id='mds1')
Server: AdminServer

Id          : mds1
Name        : MyFilter
Properties  :
     Condition :
     NOUNTYPE equals MDS_Connections AND CONTEXT user equals bruce IGNORECASE
```

## listDMSEventRoutes

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

List the events routes and their status (enabled or disabled) that are associated with the specified filter or destination. If you do not specify a filterid or destinationid, this command lists all the event routes in the Event Tracing configuration.

### Syntax

```
listDMSEventRoutes([filterid] [, destinationid][, server])
```

| Argument | Definition |
|----------|------------|
| filterid | Optional. The unique identifier for the filter. |
| destinationid | Optional. The unique identifier for the specific destination. The destination must exist. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Examples

The following example lists all event routes:

**listDMSEventRoutes()**
```
Server: AdminServer

   Filter      :   auto215443800
   Destination :   destination1
   Enabled     :   true


   Filter      :   None
   Destination :   HTTPRequestTrackerDestination
   Enabled     :   true
```

The following example lists the event routes with the filter id of filter1:

**listDMSEventRoutes(filterid='mds1')**
```
Server: AdminServer

   Filter      : mds1
   Destination : destination1
   Enabled     : false
```

The following example lists the event routes with the destination id of destination1:

**listDMSEventRoutes(destinationid='destination1'**)
```
Server: AdminServer

   Filter      : auto215443800
   Destination : destination1
   Enabled     : true
   Filter      : mds2
   Destination : destination1
```

```
Enabled      : false
Filter       : mds1
Destination  : destination1
Enabled      : false
```

## removeDMSEventDestination

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Removes an existing destination from the Event Tracing configuration. You can remove a destination only if no event route depends on the destination. If an event route that depends on the destination exists, a warning is returned.

You must be connected to the Administration Server to remove a destination. If you are not, an error is returned.

### Syntax

```
removeDMSEventDestination(id [, server])
```

| Argument | Definition |
|----------|------------|
| id | The unique identifier for the destination to be removed. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Examples

The following example removes the destination jfr:

**removeDMSEventDestination(id='jfr')**

```
Destination "jfr" removed.
```

The following example attempts to remove the destination styx.inpass.db1. However, because an event route exists for the destination, the command returns an error.

**removeDMSEventDestination(id='styx.inpass.db1')**

```
An event-route for destination 'styx.inpass.db1' exists. Unable to remove this
destination for server "AdminServer".
```

## removeDMSEventFilter

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Removes an existing filter from the Event Tracing configuration. You can remove a filter only if no event route depends on the filter. If an event route that depends on the filter exists, a warning is returned.

You must be connected to the Administration Server to remove an event filter. If you are not, an error is returned.

### Syntax

```
removeDMSEventFilter(id [, server])
```

| Argument | Definition |
|----------|------------|
| id | The unique identifier for the filter to be removed. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Example

The following example removes the filter mds1:

**removeDMSEventFilter(id='mds1')**

```
Filter "mds1" removed  for server "AdminServer".
```

The following example attempts to remove a filter for which and event-route currently exists:

**removeDMSEventFilter(id='allaccounts')**

```
Filter "allaccounts" cannot be removed. An event-route currently exists for that
filter. Remove the event-route first using the command removeDMSEventRoute().
```

## removeDMSEventRoute

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Removes the specified event route. You must be connected to the Administration Server to add an event route. If you are not, an error is returned.

### Syntax

```
removeDMSEventRoute([filterid] [, destinationid]
                    [, server])
```

| Argument | Definition |
|----------|------------|
| filterid | Optional. The unique identifier for the filter. |
| destinationid | Optional. The unique identifier for the specific destination. The destination must exist. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Example

The following example removes the event route with the filterid mds1 and the destination jfr:

**removeDMSEventRoute(filterid='mds1', destinationid='jfr')**
```
Event-route for filter "mds1", destination "jfr" removed for server "AdminServer".
```

The following example removes the event route with the destination destination1:

**removeDMSEventRoute(destinationid='destination1')**
```
Event-route for filter "None", destination "destination1" removed for server
"AdminServer".
```

## updateDMSEventDestination

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Updates an existing destination, allowing a specified argument to be updated. You must be connected to the Administration Server to update a destination. If you are not, an error is returned.

### Syntax

```
updateDMSEventDestination(id [, name,] class
                [,props= {'name': 'value'...}] [, server])
```

| Argument | Definition |
|----------|------------|
| id | The unique identifier for the destination to be updated. |
| name | Optional. A name for the destination. |
| class | The full classname of the destination.<br>See Table 6–6 for a list of available destinations. |
| props | Optional. The name/value properties to use for the destination. You can add a new property, or update or remove an existing one. If you update properties, you must specify all properties. If you omit a property, it is removed. For example, if a destination contains the properties LoggerName and severity, and you omit severity, it will be removed from the destination.<br>See addDMSEventFilter for information about the syntax and allowed values. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Examples

The following example updates the name of the destination jfr:

**updateDMSEventDestination(id='jfr', name='Alternative Flight-Recorder')**

```
Destination "jfr" updated  for server "AdminServer".
```

The following example attempts to update a destination that does not exist. The command returns an error:

**updateDMSEventDestination(id='destination1',**
**        props={'loggerName': 'MyNewTrace2-logger'})**

```
Destination "destination1" does not exist for server "AdminServer".
```

## updateDMSEventFilter

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Updates an existing filter in the Event Tracing configuration.

You must be connected to the Administration Server to update an event filter. If you are not, an error is returned.

### Syntax

```
updateDMSEventFilter(id [, name] [,etypes],
                     props= {'prop-name': 'value'...} [,server])
```

| Argument | Definition |
|----------|------------|
| id | The unique identifier for the filter to be updated. |
| name | Optional. The name of the filter to be updated. |
| etypes | Optional. A string containing a comma-separated list of event/action pairs. See addDMSEventFilter for a list of valid values. |
| props | *prop-name*: The name of the filter property. <condition> is the only valid property, and only one condition may be specified. See addDMSEventFilter for information on the syntax of *prop-name*. |
| | *value*: The value of the property of the filter. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Examples

The following example updates the filter properties for the filter with the id mds1:

```
updateDMSEventFilter(id='mds1',
 props={'condition': 'NOUNTYPE equals XYZ_Total_Connections AND CONTEXT user
equals bruce'})

Filter "mds1" updated for server "AdminServer".
.
```

The following example attempts to update a filter that does not exist:

```
updateDMSEventFilter(id='Filter2')

Filter "Filter2" does not exist for server "AdminServer".
```

## **updateDMSEventRoute**

Command Category: DMS Event Tracing

Use with WLST: Online

### Description

Enables or disables the specified event route. You must be connected to the Administration Server to update an event route. If you are not, an error is returned.

### Syntax

```
updateDMSEventRoute([filterid] [, destinationid]
        [, enable=true|false] [, server])
```

| Argument | Definition |
|----------|------------|
| filterid | Optional. The unique identifier for the filter. |
| destinationi d | Optional. The unique identifier for the specific destination. The destination must exist. |
| enable | Optional. Enables the filter. Valid values are true and false. |
| server | Optional. The server on which to perform this operation. The default is the server to which you are connected. |

### Example

The following example disables the event route with the filterid mds1 and the destinationid jfr:

**updateDMSEventRoute(filterid='mds1', destinationid='jfr', enable='false')**
Event-route for filter "mds1", destination "destination1" updated for server
"AdminServer".

# 7

# Logging Custom WLST Commands

This chapter provides detailed descriptions of custom WLST commands for logging, including command syntax, arguments and command examples. Use the logging commands to configure settings for log files and to view and search log files.

For additional details about configuring and searching log files, see "Managing Log Files and Diagnostic Data" in *Administering Oracle Fusion Middleware*.

Table 7–1 describes the different categories of logging commands.

*Table 7–1    Logging Command Categories*

| Command category | Description |
| --- | --- |
| Log Configuration Commands | Configure settings for log files, such as the level of information written to the file or the maximum file size. |
| Search and Display Commands | View Oracle Fusion Middleware log files and search log files for particular messages. |
| Selective Tracing Commands | Configure and use selective tracing, which specifies that messages are traced for specific server, loggers, or users. |

# Log Configuration Commands

Use the commands in Table 7–2 to configure settings for log files, such as the level of information written to the file or the maximum file size. In the Use with WLST column, online means the command can only be used when connected to a running server. Offline means the command can only be used when not connected to a running server. Online or offline means the command can be used in both situations.

*Table 7–2    Logging Configuration Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| configureLogHandler | Configure an existing log handler, add a new handler, or remove existing handlers. | Online |
| getLogLevel | Get the level for a given logger. | Online |
| listLoggers | Get the list of loggers and the level of each logger. | Online |
| listLogHandlers | List the configuration of one of more log handlers. | Online |
| setLogLevel | Set the level for a given logger. | Online |

## configureLogHandler

Command Category: Log Configuration

Use with WLST: Online

### Description

Configures an existing Java logging handler, adds a new handler, or removes an existing handler. It returns a java.util.List with one entry for each handler. Each entry is a javax.management.openmbean.CompositeData object describing the handler.

With this command, you can change the location of the log files, the frequency of the rotation of log files, and other log file properties.

### Syntax

```
configureLogHandler([target,] name [, maxFileSize] [,maxLogSize] [,
rotationFrequency]
 [, baseRotationTime] [, retentionPeriod] [, format] [, encoding] [, path]
 [, handlerType] [, propertyName] [, propertyValue] [, addProperty]
 [, removeProperty] [, addHandler] [, removeHandler] [, level] [, addToLogger]
 [, removeFromLogger] [, useParentHandlers]  )
```

| Argument | Definition |
|---|---|
| target | Optional. The name of a WebLogic Server instance, or a string describing a system component. For system components, refer to the component's documentation for details. |
| | The default value is the server to which WLST is connected. |
| name | Required. The name of a log handler. |
| maxFileSize | Optional. The value of the maximum file size for an ODL handler. The value is a string representing a numeric value, optionally followed by a suffix indicating a size unit (k for kilobytes, m for megabytes, g for gigabytes). |
| | If you do not specify a suffix, the value is returned in bytes. |
| | Note that this option does not apply to the QuickTrace handler. |
| maxLogSize | Optional. The value of the maximum size of the log files for an ODL handler. The value is a string representing a numeric value, optionally followed by a suffix indicating a size unit (k for kilobytes, m for megabytes, g for gigabytes). |
| | Note that this option does not apply to the QuickTrace handler. |
| rotationFrequency | Optional. The value of the rotation frequency for an ODL handler. The value is a string representing a numeric value, optionally followed by a suffix indicating a time unit (m for minutes, h for hours, d for days). The default unit is minutes. The following special values are also accepted and are converted to a numeric value in minutes: HOUR, HOURLY, DAY, DAILY, WEEK, WEEKLY, MONTH, MONTHLY. |
| | Note that this options does not apply to the QuickTrace handler. |
| baseRotationTime | Optional. The base rotation time, to be used with the rotationFrequency option. The value must be a string representing a date/time value. It can be a full date/time in ISO 8601 date/time format, or a short form including only hours and minutes. The default baseRotationTime is 00:00. |
| | Note that this option does not apply to the QuickTrace handler. |

| Argument | Definition |
|---|---|
| retentionPeriod | Optional. The amount of time that the log file is retained. The value must be a string representing a numeric value, optionally followed by a suffix indicating a time unit (m for minutes, h for hours, d for days). The default unit is minutes. The following special values are also accepted and are converted to a numeric value in minutes: HOUR, HOURLY, DAY, DAILY, WEEK, WEEKLY, MONTH, MONTHLY.<br><br>Note that this option does not apply to the QuickTrace handler. |
| format | Optional. The format for the ODL handler. Valid values are one of the following strings: "ODL-Text" or "ODL-XML". The default format is ODL-Text. |
| encoding | Optional. The character encoding for the log file. |
| path | Optional. The log file path.<br><br>Note that this option does not apply to the QuickTrace handler. |
| handlerType | Optional. The name of the Java class that provides the handler implementation. It must be an instance of java.util.logging.Handler or oracle.core.ojdl.logging.HandlerFactory. |
| propertyName | Optional. The name of an advanced handler property to be added or updated. The property value is specified with the propertyValue option. See the documentation for the handler for valid properties. |
| propertyValue | Optional. The new value for the handler property defined by the propertyName option. |
| addProperty | Optional. A Jython boolean value. Used in conjunction with the propertyName and propertyValue options to define that a new property is to be added to the handler. |
| removeProperty | Optional. A list of one or more handler properties to be removed. |
| addHandler | Optional. A boolean value. If the value is true, then the named handler will be added. |
| removeHandler | Optional. A boolean value. If the value is true, then the named handler is removed. |
| level | Optional. A Java or ODL level value. The handler level will be set to the given level. |
| addToLogger | Optional. A list of logger names. The handler is added to the given logger names. |
| removeFromLogger | Optional. A list of logger names. The handler is removed from the given loggers. |
| useParentHandlers | Optional. A boolean value. Sets the useParentHandlers flag on the loggers defined by the addToLogger or removeFromLogger options. |

The following table lists the properties for the quicktrace-handler. This handler allows you to trace messages from specific loggers and store the messages in memory. For more information, see "Configuring QuickTrace" in *Administering Oracle Fusion Middleware.*

| QuickTrace Property | Description |
|---|---|
| bufferSize | The approximate size of the circular QuickTrace buffer, in which log records are stored in memory. Note that actual memory consumption may be less than, but not more than this value. |
| enableDMSMetrics | If specified as true, DMS metrics are enabled for the quicktrace-handler. The default is true. |

| QuickTrace Property | Description |
| --- | --- |
| enableUserBuffer | If specified as true, the handler maintains an individual buffer for each user specified in the reserveBufferUserID property. If the user is not defined in the reserveBufferUserID property, the messages are cached in the COMMON buffer. |
| | If specified as false, the handler maintains only one buffer, COMMON. |
| | The default is false. |
| flushOnDump | If specified as true, the buffer is flushed when you execute the executeDump command. The default is true. |
| includeMessageArguments | If specified as true, message arguments are included with the formatted log messages that have a message ID. The default is false. |
| maxFieldLength | The maximum length, in bytes, for each field in a message. The fields can include the message text, supplemental attributes, thread name, source class name, source method name, and message arguments. |
| | The default is 240 bytes. |
| | A small number can restrict the amount of information returned for a message. An excessively number can reduce the amount of log records in the buffer because each message uses more bytes. |
| reserveBufferUserID | A list of user IDs, separated by a comma. If enableUserBuffer is specified as true, any log messages related to the user are written to a separate buffer. |
| supplementalAttributes | A list of supplemental attribute names. The attributes are listed in the logging.xml file. |
| | Setting supplemental attributes requires additional memory or CPU time. |
| useDefaultAttributes | If specified as true, default attribute values are added to each log message. The default attributes are HOST_ID, HOST_NWADDR, and USER_ID. |
| useLoggingContext | If specified as true, the log message includes DMS logging context attributes. The default is false. |
| | If you enable this option, the trace requires additional CPU time. |
| useRealThreadID | If specified as true, the handler attempts to use the real thread ID instead of the thread ID that is provided by the jave.util.logging.logRecord. The default is false. |
| | If you enable this option, the trace requires additional CPU time. |
| useThreadName | If specified as true, the log message includes the thread name instead of the thread ID. The default is false. |

## Examples

The following example specifies the maximum file size for the odl-handler:

```
configureLogHandler(name="odl-handler", maxFileSize="5M")
```

The following example specifies the rotation frequency for the odl-handler:

```
configureLogHandler(name="odl-handler", rotationFrequency="daily")
```

The following example specifies the rotation frequency and the retention period for the odl-handler. It also removes the properties maxFileSize:

```
configureLogHandler(name="odl-handler", rotationFrequency="daily",
```

```
                        retentionPeriod="week", removeProperty='maxFileSize'])
```

The following example configures the quicktrace-handler, adding the logger oracle.adf.faces, and enabling user buffers for user1 and user2:

```
configureLogHandler(name="quicktrace-handler", addToLogger="oracle.adf.faces",
        propertyName="enableUserBuffer", propertyValue="true",
      propertyName="reserveBufferUserID", propertyValue="user1, user2")
```

The oracle.adf logger is associated with the handlers odl-handler, wls-domain, and console-handler. When you set the level of the logger, these handlers will use the same level (TRACE:1) for the logger oracle.adf. As a result, much information will be written to the log files, consuming resources. To avoid consuming resources, set the level of the handlers to a lower level, such as WARNING or INFORMATION. For example:

```
configureLogHandler(name="odl-handler", level="WARNING:1")
configureLogHandler(name="wls-domain", level="WARNING:1")
configureLogHandler(name="console-handler", level="WARNING:1")
```

## getLogLevel

Command Category: Log Configuration

Use with WLST: Online

### Description

Returns the level of a given Java logger.

The returned value is a string with the logger's level, or None if the logger does not exist. An empty string indicates that the logger level is null.

### Syntax

```
getLogLevel( [target,] logger [, runtime] )
```

| Argument | Definition |
|----------|------------|
| target | Optional. The name of a WebLogic Server instance, or a string describing a system component. For system components, refer to the component's documentation for details. |
| | The default value is the server to which WLST is connected. |
| logger | A logger name. An empty string denotes the root logger. |
| | This option is required and has no default. |
| runtime | Optional. A Jython boolean value (0 or 1) that determines if the operation is to list runtime loggers or config loggers. The default value is 1 (runtime). |

### Examples

The following example returns the level for the logger oracle:

**getLogLevel(logger='oracle')**
NOTIFICATION:1

The following example returns the level for the logger oracle, specifying only config loggers, not runtime loggers:

**getLogLevel(logger='oracle', runtime=0)**
NOTIFICATION:1

The following example returns the level for the logger oracle on the Oracle WebLogic Server server2:

**getLogLevel(logger='oracle', target='server2')**
NOTIFICATION:1

## listLoggers

Command Category: Log Configuration

Use with WLST: Online

### Description

Lists Java loggers and their levels. The command returns a PyDictionary object where the keys are logger names and the associated values are the logger levels. An empty level is used to indicate that the logger does not have the level set.

### Syntax

```
listLoggers([target] [, pattern] [,runtime])
```

| Argument | Definition |
| --- | --- |
| target | Optional. The name of a WebLogic Server instance, or a string describing a system component. For system components, refer to the component's documentation for details. |
| | The default value is the server to which WLST is connected. |
| pattern | Optional. The name of a WebLogic Server instance, or a string describing a system component. For system components, refer to the component's documentation for details. |
| runtime | Optional. A Jython boolean value (0 or 1) that determines if the operation is to list runtime loggers or config loggers. The default value is 1 (runtime). |

### Examples

The following example lists all of the loggers:

**listLoggers()**

The following example lists all of the loggers that start with the name oracle.*.

**listLoggers(pattern="oracle.*")**

The following example list all config loggers:

**listLoggers(runtime=0)**

The following example list all loggers for the WebLogic Server server1:

**listLoggers(target="server1")**

## listLogHandlers

Command Category: Log Configuration

Use with WLST: Online

### Description

Lists Java log handlers configuration. This command returns a java.util.List with one entry for each handler. Each entry is a javax.management.openmbean.CompositeData object describing the handler.

### Syntax

```
listLogHandlers([target] [, name])
```

| Argument | Definition |
|----------|------------|
| target | Optional. The name of a WebLogic Server instance, or a string describing a system component. For system components, refer to the component's documentation for details. |
|  | The default value is the server to which WLST is connected. |
| name | Optional. The name of a log handler. If the name is not provided, then all handlers are listed. |

### Examples

The following example lists all log handlers:

**listLogHandlers()**

The following example lists all log handlers named odl-handler:

**listLogHandlers(name="odl-handler")**

The following example lists all log handlers for the WebLogic Server server1:

**listLogHandlers(target="server1")**

## setLogLevel

Command Category: Log Configuration

Use with WLST: Online

### Description

Sets the level of information written by a given Java logger to a log file.

### Syntax

```
setLogLevel([target,] logger [, addlogger] , level [, runtime] [, persist] )
```

| Argument | Definition |
|---|---|
| target | Optional. The name of a WebLogic Server instance, or a string describing a system component. For system components, refer to the component's documentation for details. |
| | The default value is the server to which WLST is connected. |
| logger | A logger name. An empty string denotes the root logger. |
| | This option is required and has no default. The command throws an exception if the logger does not exist, unless the addLogger option is also used. |
| addLogger | Optional. A Jython boolean value (0 or 1) that determines if the logger should be created if it does not exist. This option is deprecated for runtime mode. Adding a runtime logger may have no effect because the logger may be garbage collected. If you need to set the level for a logger that has not yet been created, use the persist mode. |
| level | The level name. It can be either a Java level or an ODL level. Some valid Java levels are: SEVERE, WARNING, INFO, CONFIG, FINE, FINER, OR FINEST. Valid ODL levels include a message type followed by a colon and a message level. The valid ODL message types are: INCIDENT_ERROR, ERROR, WARNING, NOTIFICATION, TRACE, and UNKNOWN. The message level is represented by an integer value that qualifies the message type. Possible values are from 1 (highest severity) through 32 (lowest severity). |
| | An empty string can be used to set the level to null (inherited from parent). |
| | This option is required; there is no default value. |
| runtime | Optional. A Jython boolean value (0 or 1) that determines if the operation is to list runtime loggers or config loggers. The default value is 1 (runtime). If the target is a system component that does not support changing runtime loggers, this option is ignored. |
| | Note: Because runtime loggers may be garbage collected, you should change the level of the runtime logger only if you know that the logger exists and that there is a strong reference to the logger. If the logger is garbage collected, any changes made to the logger level in runtime mode that are not persisted may be lost. |
| persist | Optional. A Jython boolean value (0 or 1) that determines if the level should be saved to the configuration file. A value of 0 specifies that the level will be saved; a value of 1 that it will not. The default value is 1. |

### Examples

The following example sets the log level to NOTIFICATION:1 for the logger oracle.my.logger:

```
setLogLevel(logger="oracle.my.logger", level="NOTIFICATION:1")
```

The following example sets the log level to TRACE:1 for the logger oracle.my.logger and specifies that the level should be saved to the configuration file:

```
setLogLevel(logger="oracle.my.logger", level="TRACE:1", persist=0)
```

The following example sets the log level to WARNING for the config logger oracle.my.logger on the WebLogic Server server1:

```
setLogLevel(target="server1", logger="oracle.my.logger", level="WARNING",
runtime=0)
```

## Search and Display Commands

Use the commands in Table 7–3 to view Oracle Fusion Middleware log files and to search log files for particular messages.

***Table 7–3    Search and Display Commands***

| Use this command... | To... | Use with WLST... |
|---|---|---|
| displayLogs | List the logs for one or more components. | Online or Offline |
| listLogs | Search and display the contents of log files. | Online or Offline |

# displayLogs

Command Category: Search and Display

Use with WLST: Online or Offline

## Description

Search and display the contents of diagnostic log files. The command returns a value only when the returnData option is set to true. By default it will not return any data. The return value depends on the option used.

## Syntax

```
displayLogs([searchString,][target] [, oracleInstance] [, log] [, last] [, tail]
[, pattern] [, ecid] [, component] [, module] [, type] [, app] [, query] [,
groupBy]
[, orderBy [, returnData] [, format] [, exportFile] [, follow])
```

| Argument | Definition |
|---|---|
| searchString | An optional search string. Only messages that contain the given string (case-insensitive) will be returned. |
| | Note that the displayLogs command can read logs in multiple formats and it converts the messages to ODL format. The search will be performed in the native format, if possible. Otherwise, it may be performed in the message contents, and it may exclude mark-up. Therefore you should avoid using mark-up characters in the search string. |
| target | Optional. The name of a WebLogic Server instance, or a system component. |
| | For a system component, the syntax for the target is: |
| | sc:*component-name* |
| | In connected mode, the default target is the WebLogic domain. In disconnected mode, there is no default; the target option is required. |
| oracleInstance | Optional. Defines the path to the ORACLE_INSTANCE or WebLogic domain home. The command is executed in disconnected mode when you use this parameter. |
| log | Optional. A log file path. The command will read messages from the given log file. If the log file path is not given, the command will read all logs associated with the given target. |
| last | Optional. An integer value. Restricts the search to messages logged within the last minutes. The value can have a suffix s (second), m (minute), h (hour), or d (day) to specify a different time unit. (For example, last='2h' will be interpreted as the last 2 hours). |
| tail | Optional. An integer value. Restrict the search to the last *n* messages from each log file and limits the number of messages displayed to *n*. |
| pattern | Optional. A regular expression pattern. Only messages that contain the given pattern are returned. Using the pattern option is similar to using the searchString argument, except that you can use a regular expression. |
| | The regular expression pattern search is case sensitive (unless you explicitly turn on case-insensitive flags in the pattern). The pattern must follow java.util.regex syntax. |
| ecid | Optional. A string or string sequence containing one or more Execution Context ID (ECID) values to be used as a filter for log messages. |
| component | Optional. A string or string sequence containing one or more component ID values to be used as a filter for log messages. |

| Argument | Definition |
|----------|------------|
| module | Optional. A string or string sequence containing one or more module ID values to be used as a filter for log messages. |
| type | Optional. A string or string sequence containing one or more message type values to be used as a filter for log messages. |
| app | Optional. A string or string sequence containing one or more application values to be used as a filter for log messages. |
| query | Optional. A string that specifies an expression used to filter the contents of log messages.<br><br>A simple expression has the form:<br><br>*field-name operator value*<br><br>where *field-name* is a log record field name and *operator* is an appropriate operator for the field type (for example, you can specify equals, startsWith, contains or matches for string fields).<br><br>A field name is either one of the standard ODL attribute names (such as COMPONENT_ID, MSG_TYPE, MSG_TEXT, and SUPPL_DETAIL), or the name of a supplemental attribute (application specific), prefixed by SUPPL_ATTR. (For example, SUPPL_ATTR.myAttribute).<br><br>A few common supplemental attributes can be used without the prefix. For example, you can use APP to filter by application name.<br><br>You can combine multiple simple expressions using the boolean operators and, or and not to create complex expressions, and you can use parenthesis for grouping expressions.<br><br>See *Administering Oracle Fusion Middleware* for a detailed description of the query syntax. |
| groupBy | Optional. A string list. When the groupBy option is used, the output is a count of log messages, grouped by the attributes defined in the string list. |
| orderBy | Optional. A string list that defines the sort order for the result. The values are log message attribute names. The name may be extended with an optional suffix :asc or :desc to specify ascending or descending sorting. The default sort order is ascending.<br><br>By default, the result is sorted by time. |
| returnData | Optional. A Jython boolean value (0 or 1). If the value is true the command will return data (for example, to be used in a script). The default value is false, which means that the command only displays the data but does not return any data. |
| format | Optional. A string defined the output format. Valid values are ODL-Text, ODL-XML, ODL-complete and simple. The default format is ODL-Text. |
| exportFile | Optional. The name of a file to where the command output is written. By default, the output is written to standard output. |
| follow (f) | Optional. Puts the command in "follow" mode so that it continues to read the logs and display messages as new messages are added to the logs (similar to the UNIX tail -f command). The command will not return when the f option is used. This option is currently not supported with system components. |

## Examples

The following example displays the last 100 messages from all log files in the domain:.

**displayLogs(tail=100)**

The following example displays all messages logged in the last 15 minutes:

```
displayLogs(last='15m')
```

The following example displays log messages that contain a given string:

```
displayLogs('Exception')
```

The following example displays log messages that contain a given ECID:

```
displayLogs(ecid='0000Hl9TwKUCslT6uBi8UH18lkWX000002')
```

The following example displays log messages of type ERROR or INCIDENT_ERROR:

```
displayLogs(type=['ERROR','INCIDENT_ERROR'])
```

The following example displays log messages for a given Java EE application:

```
displayLogs(app="myApplication")
```

The following example displays messages for a system component, ohs1:

```
displayLogs(target="sc:ohs1")
```

The following example displays a message summary by component and type:

```
displayLogs(groupBy=['COMPONENT_ID', 'MSG_TYPE'])
```

The following example displays messages for a particular time interval:

```
displayLogs(query="TIME from 11:15 and TIME to 11:20")
```

The following example shows an advanced query:

```
displayLogs(query="TIME from 11:15 and TIME to 11:20 and ( MSG_TEXT contains
    exception or SUPPL_DETAIL contains exception )")
```

A similar query could be written as:

```
displayLogs("exception", query="TIME from 11:15 and TIME to 11:20")
```

## listLogs

Command Category: Search and Display

Use with WLST: Online or Offline

### Description

Lists log files for Oracle Fusion Middleware components. This command returns a PyArray with one element for each log. The elements of the array are javax.management.openmbean.CompositeData objects describing each log.

### Syntax

```
listLogs([target] [, oracleInstance] [, unit] [, fullTime]
```

| Argument | Definition |
|---|---|
| target | Optional. The name of a WebLogic Server instance, or an Oracle Fusion Middleware system component.<br><br>For a system component, the syntax for the target is:<br><br>sc:*component-name*<br><br>In connected mode, the default target is the WebLogic domain.<br><br>In disconnected mode, there is no default; the target option is required. |
| oracleInstance | Optional. Defines the path to the ORACLE_INSTANCE or WebLogic domain home. The command is executed in disconnected mode when you use this parameter. |
| unit | Optional. Defines the unit to use for reporting file size. Valid values are B (bytes), K (kilobytes), M (megabytes), G (gigabytes), or H (display size in a human-readable form, similar to the UNIX ls -h option). The default value is H. |
| fullTime | Optional. A Jython Boolean value. If true, reports the full time for the log file last modified time. Otherwise, it displays a short version of the time. The default value is false. |

### Examples

The following example lists all of the log files for the WebLogic domain:

**listLogs()**

The following example lists the log files for the WebLogic Server server1:

**listLogs(target="server1")**

The following example lists the log files for the Oracle HTTP Server ohs1:

**listLogs(target="sc:ohs1"**)

The following example, used in disconnected mode, lists the log files for the WebLogic Server server1:

**listLogs(oracleInstance="/scratch/Oracle/domains/base_domain",**
**target="server1")**

## Selective Tracing Commands

Use the commands in Table 7–4 to configure and use selective tracing. Selective tracing provides fine-grained logging for specified users or other properties of a request. In the Use with WLST column, online means the command can only be used when connected to a running server.

*Table 7–4    Tracing Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| configureTraceProvider | Configure a trace provider. | Online |
| configureTracingLoggers | Configure one or more loggers for selective tracing. | Online |
| listActiveTraces | List the active traces. | Online |
| listTraceProviders | List the tracing providers. | |
| listTracingLoggers | List the loggers that support selective tracing. | Online |
| startTracing | Start a selective tracing sessions. | Online |
| stopTracing | Stop one or more selective tracing sessions. | Online |

## configureTraceProvider

Command Category: Tracing

Use with WLST: Online

### Description

Configures a trace provider. Currently, the only available option is to enable or disable the provider.

### Syntax

```
configureTraceProvider([target,] name, action)
```

| Argument | Definition |
|----------|------------|
| target | Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, the targets are all running server instances in the domain that are JRF-enabled. |
| name | Required. The trace provider name. |
| action | Enables or disables tracing for the provider. Valid values are enable and disable. This option is required; there is no default value. |

### Examples

The following example disables the DMS trace provider on all running servers in the domain:

```
configureTraceProvider(name='DMS', action='disable')
```

The following example enables the DMS trace provider for the server wls_server1:

```
configureTraceProvider(target='wls_server1', name='DMS', action='enable')
```

## configureTracingLoggers

Command Category: Tracing

Use with WLST: Online

### Description

Configures one or more loggers for selective tracing. This command also enables or disables a logger for selective tracing.

### Syntax

```
configureTracingLoggers([target] [, pattern,] action)
```

| Argument | Definition |
|----------|-----------|
| target | Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled will be configured for tracing. |
| pattern | Optional. A regular expression pattern that is used to filter logger names. The default value matches all tracing logger names. |
| action | Required. Enables or disables all loggers for tracing. Valid values are enable and disable. There is no default value. |

### Examples

The following example configures selective tracing for all loggers beginning with `oracle.security`:

**configureTracingLoggers(pattern='oracle.security.*', action="enable")**
Configured 80 loggers

The following example disables selective tracing for all loggers:

**configureTracingLoggers(action="disable")**
Configured 969 loggers

## listActiveTraces

Command Category: Tracing

Use with WLST: Online

### Description

Lists the active traces.

### Syntax

```
listActiveTraces([target])
```

| Argument | Definition |
|----------|------------|
| target | Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled are listed. |

### Example

The following example lists the active traces:

**listActiveTraces()**
```
-----------------------------------+---------+----------+------+------------------+---
Trace ID                           |Attr. Name|Attr. Value| Level| Start Time      |Exp.
Time
-----------------------------------+---------+----------+------+------------------+---
bf13025b-b8f8-480d-8d92-14200a669b3e |USER_ID  |user1     | FINE |  5/28/13 12:28 PM |
a04b47f7-2830-4d80-92ee-ba160cdacf6b |USER_ID  |user2     | FINE |  5/28/13 12:30 PM |
```

## listTraceProviders

Command Category: Tracing

Use with WLST: Online

### Description

Lists the name, status, description and supported parameters for the available trace providers. The status of a provider can be either `enabled` meaning that the provider is enabled on all targets, `disabled` meaning that the provider is disabled on all targets, or `mixed` meaning that the provider is enabled on some targets.

### Syntax

```
listTraceProviders([target,] [name])
```

| Argument | Definition |
| --- | --- |
| target | Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, providers on all running server instances in the domain that are JRF-enabled will be listed. |
| name | Optional. A trace provider name. If you specify this parameter, only this provider is listed. |

### Example

The following example lists all trace providers for all running servers in the domain:

**listTraceProviders()**

## listTracingLoggers

Command Category: Tracing

Use with WLST: Online or Offline

### Description

Lists the loggers that support selective tracing. This command displays a table of logger names and their tracing status. The status `enabled` means that the logger is enabled for tracing on all servers. The status `disabled` means that the logger is disabled for tracing on all servers. The status `mixed` means that the logger is enabled for tracing on some servers, but disabled on others.

### Syntax

```
listTracingLoggers([target] [, pattern])
```

| Argument | Definition |
|----------|------------|
| target | Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled are listed. |
| pattern | Optional. A regular expression pattern that is used to filter logger names. The default value matches all tracing logger names. |

### Example

The following example lists all tracing loggers beginning with `oracle.security`:

**listTracingLoggers(pattern="oracle.security.*")**
```
-----------------------------------------------------------------+--------
Logger                                                           | Status
-----------------------------------------------------------------+--------
oracle.security                                                  | enabled
oracle.security.audit.logger                                     | enabled
oracle.security.audit.config                                     | enabled
 .
 .
 .
```

## startTracing

Command Category: Tracing

Use with WLST: Online

### Description

Starts a new selective tracing session for a specified user or DMS context attribute at a specified level of tracing.

### Syntax

```
startTracing([target,] [ traceId,] [attrName, attrValue,] [user,] level [, desc])
```

| Argument | Definition |
|----------|------------|
| target | Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled are included in the trace. |
| traceId | Optional. An identifier for the tracing session. If a traceId is not provided, the command generates a unique traceId. |
| attrName | Optional, unless the user argument is not specified. Valid values are USER _ID, APP, CLIENT_HOST, CLIENT_ADDR, composite_name, WEBSERVICE.name, WEBSERVICE_PORT.name. |
| attrValue | Required if attrName is specified. The value of the attribute. |
| user | The user name. Messages associated with the user are returned. This is equivalent to passing the USER_ID with the attrName and AttrValue options. |
| level | Required. The tracing level. The level must be a valid Java or ODL level. See the table "Mapping of Log Levels Among ODL, Oracle WebLogic Server, and Java" in *Administering Oracle Fusion Middleware*. |
| desc | Optional. A description of the tracing session. |

### Example

The following example starts a trace for messages associated with user1 and sets the level of information to FINE:

**startTracing(user="user1",level="FINE")**
```
Started tracing with ID: 885649f7-8efd-4a7a-9898-accbfc0bbba3
```

## stopTracing

Command Category: Tracing

Use with WLST: Online

### Description

Stops one or more selective tracing sessions.

### Syntax

```
stopTracing([target,] {stopAll] | traceId | attrName, attrValue | user} [,
createIncident)
```

| Argument | Definition |
|----------|------------|
| target | Optional. The name of a WebLogic Server instance, or an array of strings containing one or more target names. By default, loggers on all running server instances in the domain that are JRF-enabled are included in the operation. |
| stopAll | A Jython boolean value (0 or 1) that determines if all of the active traces are stopped. Required if the traceId, user, or attrName and attrValue arguments are not specified. The default value is 0 (false). |
| traceId | An identifier for the tracing session to be stopped. Required if the stopAll, user, or attrName and attrValue arguments are not specified. |
| attrName | Valid values are USER _ID, APP, CLIENT_HOST, CLIENT_ADDR, composite_name, WEBSERVICE.name, WEBSERVICE_PORT.name. Required if the traceId, user, stopAll arguments are not specified. |
| attrValue | Required if attrName is specified. The value of the attribute. |
| user | The user name. All tracing sessions associated with the user are stopped. Required if the stopAll, traceId, or attrName and attrValue arguments are not specified. |
| createIncident | Optional. A Jython boolean value (0 or 1). If true, an incident will be created for each trace that is stopped. The default value is 0 (false). |

### Examples

The following example stops a tracing session with a specified traceId:

**stopTracing(traceId="a04b47f7-2830-4d80-92ee-ba160cdacf6b")**
```
Stopped 1 traces
```

The following example stops all tracing sessions:

**stopTracing(stopAll=1)**
```
Stopped 1 traces
```

# 8

# Diagnostic Framework Custom WLST Commands

The Diagnostic Framework aids in capturing relevant and timely diagnostics for critical errors. The diagnostics can be sent to Oracle Support for further analysis. Use the Diagnostic Framework commands to generate incidents, query existing incidents and execute individual diagnostics dumps to gather specific diagnostics data. This chapter provides detailed descriptions of WLST commands for the Diagnostic Framework, including command syntax, arguments and command examples.

For additional information about using the Diagnostic Framework, see "Diagnosing Problems" in *Administering Oracle Fusion Middleware.*

Table 8–1 lists the different categories of Diagnostic Framework commands.

*Table 8–1    Diagnostic Command Categories*

| Command Category | Description |
| --- | --- |
| Incident Commands | View problems and incidents and create incidents. |
| Diagnostic Dump Commands | Display information about dumps and execute dumps. |
| Dump Sampling Commands | Capture samples of diagnostic dumps at specified intervals.s |

# Incident Commands

Use the commands in Table 8–2 to view problems and incidents and to create incidents.

*Table 8–2    Incident Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| createAggregatedIncident | Create an aggregated incident, containing zip files that contain copies of incidents that match the specified criteria. | Online |
| createIncident | Create a diagnostic incident. | Online |
| getIncidentFile | Retrieve the contents of the specified incident file. | Online |
| listADRHomes | List the set of ADR Home paths. | Online |
| listIncidents | List a set of diagnostic incidents. | Online |
| listProblems | List a set of diagnostic problems. | Online |
| queryIncidents | List the incidents that meet the specified criteria. | Online |
| reloadCustomRules | Reload all custom diagnostic rules or the specified rule. | Online, Offline |
| showIncident | Show the details of a specified incident. | Online |

## createAggregatedIncident

Use with WLST: Online

### Description

Creates an aggregated incident, containing zip files that contain copies of incidents that match the specified criteria.

### Syntax

```
createAggregatedIncident(query [, servers])
```

| Argument | Definition |
|----------|------------|
| query | An expression composed of simple expressions, which can be connected by Boolean operators. An expression contains an incident attribute, an operator, and a string, in the following format: |
| | *attribute operator* "*string*" |
| | Simple expressions can be connected by the Boolean operators AND or OR, grouped by parentheses () |
| | The following incident attributes are supported: |
| | ■ TIMESTAMP: Incident creation time. You can use the `from` and `to` operators to specify a time range. The date format is `YYYY-MM-DD HH:MM`. |
| | ■ ECID: Execution Context ID |
| | ■ PROBLEM_KEY: Problem Key |
| | ■ MSG_FACILITY: The error message facility, such as ORA or OHS. |
| | ■ MSG_NUMBER: The error message ID, such as 600. |
| | Custom incident attributes are also supported. For example, TRACEID, APP, URI, AND DSID are supported. |
| | The following operators are supported: |
| | ■ equals |
| | ■ notEqual |
| | ■ startsWith |
| | ■ endsWith |
| | ■ contains |
| | ■ isNull |
| | ■ notNull |
| servers | The name of one or more servers to query. This argument is optional. If you do not specify it, the command operates on all servers in the domain. |
| | This option is only valid when you are connected to the Administration Server. |

### Examples

The following example creates an aggregated incident for al incidents that contain the ODL_TRACE_ID of 123456 on the server wls_server1:

```
createAggregatedIncident(query="ORDL_TRACE_ID equals 123456", servers="wls_
server1")
Incident 55 created, containing the following incidents:
```

```
Server wls_server1
Incident Id    Problem Key                                Incident Time
15                 TRACE [123456] [MANUAL]        Mon Apr 15 11:22:12 EDT 2013
```

The following example creates an aggregated incident for al incidents that contain the ODL_TRACE_ID of 123456 on all servers in the domain:

```
createAggregatedIncident(query="ORDL_TRACE_ID equals 123456", servers="wls_
server1")
Incident 55 created, containing the following incidents:
Server wls_server1, wls_server2
Incident Id    Problem Key                                Incident Time
15                 TRACE [123456] [MANUAL]        Mon Apr 15 11:22:12 EDT 2013
```

## createIncident

Use with WLST: Online

### Description

Creates a diagnostic incident, using the specified information to determine the set of diagnostic rules and actions to execute.

### Syntax

```
createIncident([adrHome] [,incidentTime] [,messageId] [,ecid] [,appName]
  [,description] [,server])
```

| Argument | Definition |
| --- | --- |
| adrHome | The path for the ADR Home in which to create the incident. The ADR Home must exist. If this argument is not specified, the default ADR Home is used. |
| | The default ADR Home is the following location: |
| | *ADR_BASE*/diag/OFM/*domain_name*/*server_name* |
| incidentTime | The timestamp at which the incident occurred. If this not specified, the current time is used. You can specify the following: |
| | ■ The time of the current day, in the format HH:MM. For example: 19:45 |
| | ■ The date and time, in the format MM/DD/YYYY HH:MM |
| messageId | The ID of the error message. For example, MDS-50400. |
| ecid | The Execution Context ID for the error message. |
| appNname | The name of the deployed application for which the diagnostics are being gathered. |
| | For example, if you have multiple ADF applications deployed, each may register a dump called adf.dump. To execute this command for a specific application, you must specify the application name. |
| description | Descriptive text to associate with the incident. This is useful when reviewing the incident at a later time. |
| server | The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server. |

### Example

The following example creates an incident that is related to messages with the ID MDS-50400:

```
createIncident(messageId="MDS-50400", description="sample incident")
Incident Id: 3
Problem Id: 2
Problem Key: MDS-50400 [MANUAL]
Incident Time:Tue May 28 11:52:45 PDT 2013
Error Message Id: MDS-50400
Execution Context:null
Flood Controlled: false
Dump Files :
   jvm_threads25_i3.txt
   dms_metrics26_i3.txt
```

```
dfw_samplingArchive28_i3.readme.txt
odl_logs29_i3.txt
```

## getIncidentFile

Use with WLST: Online

### Description

Retrieves the contents of the specified incident file.

### Syntax

getIncidentFile(id, name [,outputFile] [,adrHome] [,server])

| Argument | Definition |
|----------|------------|
| id | The ID of the incident that you want to retrieve. |
| name | The name of the file to retrieve. To find the name of the file, use the showIncident command. |
| outputFile | The name of the file to which to write the output. |
| adrHome | The path for the ADR Home from which to retrieve the information. If this argument is not specified, the default ADR Home will be queried. |
| | The default ADR Home is the following location: |
| | *ADR_BASE*/diag/OFM/*domain_name*/*server_name* |
| server | The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server. |

### Example

The following example writes the contents of the incident dms_metrics3_i1.dmp to the specified output file:

**getIncidentFile(id='1', name='dms_metrics3_i1.dmp', outputFile='/tmp/incident1_dms.txt')**
The content of 'dms_metrics3_i1.dmp'is written to /tmp/incident1_dms.txt

## listADRHomes

Use with WLST: Online

### Description

Lists the paths of all of the ADR Homes for the server.

### Syntax

```
listADRHomes([server])
```

| Argument | Definition |
| --- | --- |
| server | The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server. |

### Example

The following example lists the paths of the ADR homes:

**listADRHomes()**
```
diag/ofm/base_domain/AdminServer
diag/ofm/EMGC_DOMAIN/EMOMS
```

## listIncidents

Use with WLST: Online

### Description

Lists the set of diagnostic incidents for the given problem ID, if specified, or all available incidents.

### Syntax

```
listIncidents([id] [, adrHome] [,server])
```

| Argument | Definition |
|----------|------------|
| id | The ID of the problem for which you want to list the set of diagnostic incidents. |
| adrHome | The path for the ADR Home from which to query incidents. If this argument is not specified, the default ADR Home will be queried. |
|  | The default ADR Home is the following location: |
|  | *ADR_BASE*/diag/OFM/*domain_name*/*server_name* |
| server | The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server. |

### Example

The following example lists the incidents associated with the problem with the ID 1:

```
listIncidents(id="1")
Incident Id    Incident Time                 Problem Key
      2        Tue May 28 11:05:59 PDT 2013   MDS-50500 [MANUAL]
      1        Tue May 28 11:02:22 PDT 2013   MDS-50500 [MANUAL]
```

## listProblems

Use with WLST: Online

## Description

Lists the set of diagnostic problems associated with the specified ADR Home.

## Syntax

```
listProblems([adrHome][,server])
```

| Argument | Definition |
| --- | --- |
| adrHome | The path for the ADR Home from which to query problems. If this argument is not specified, the default ADR Home will be queried. |
| | The default ADR Home is the following location: |
| | *ADR_BASE*/diag/OFM/*domain_name*/*server_name* |
| server | The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server. |

## Example

The following example lists the diagnostic problems in the default ADR home:

```
listProblems()
Problem Id      Problem Key
        1       MDS-50500 [MANUAL]
        2       JOC-38922 [AdminServer] [oracle.cache.network]
```

## queryIncidents

Use with WLST: Online

### Description

Lists the incidents that meet the specified criteria. You can query for the value of particular attributes across one or more servers, or all servers in a domain.

### Syntax

```
queryIncidents(query [,servers])
```

| Argument | Definition |
| --- | --- |
| query | An expression composed of simple expressions, which can be connected by Boolean operators. An expression contains an incident attribute, an operator, and a string, in the following format: |
| | *attribute operator* "*string*" |
| | Simple expressions can be connected by the Boolean operators AND or OR, grouped by parentheses () |
| | The following incident attributes are supported: |
| | ■ TIMESTAMP: Incident creation time. You can use the `from` and `to` operators to specify a time range. The date format is `YYYY-MM-DD HH:MM`. |
| | ■ ECID: Execution Context ID |
| | ■ PROBLEM_KEY: Problem Key |
| | ■ MSG_FACILITY: The error message facility, such as ORA or OHS. |
| | ■ MSG_NUMBER: The error message ID, such as 600 |
| | Custom incident attributes are also supported. For example, TRACEID, APP, URI, AND DSID are supported. In addition, the context values, as shown in the incident readme.txt file, are supported. For example, DFW_APP_NAME and DFW_USER_NAME are supported. |
| | The following operators are supported: |
| | ■ equals |
| | ■ notEqual |
| | ■ startsWith |
| | ■ endsWith |
| | ■ contains |
| | ■ isNull |
| | ■ notNull |
| servers | The name of one or more servers to query. This argument is optional. If you do not specify it, the command operates on all servers in the domain. |
| | This option is only valid when you are connected to the Administration Server. |

### Examples

The following example queries all incidents in the domain for the ECID f19wAgN000001:

```
queryIncidents(query="ECID equals f19wAgN000001")
```

The following example queries all incidents that occurred between April 1, 2013 and April 15, 2013, for the server wls_server1:

```
 queryIncidents(query="TIMESTAMP from '2013-04-01 00:00'AND TIMESTAMP to
'2013-04-15 00:00'", servers=["wls_server1"])
```

## reloadCustomRules

Use with WLST: Online, Offline

### Description

Reloads all custom diagnostic rules or the specified custom diagnostic rule.

### Syntax

```
reloadCustomRules([name] [, server])
```

| Argument | Definition |
|----------|------------|
| name | The name of a custom diagnostic rule. This argument is optional. If you specify it, only the named rule is reloaded. If you do not specify this argument, all custom diagnostic rules are reloaded. |
| | The file containing the custom diagnostic rule must be located in one of the following directories: |
| | *DOMAIN_HOME*/config/fmwconfig/dfw<br>*DOMAIN_HOME*/config/fmwconfig/servers/*server_name*/dfw |
| server | The name of the server to which to reload the rules. This argument is optional. If you do not specify it, the rules are reloaded to all servers. |
| | This option is only valid when you are connected to the Administration Server. |

### Example

The following example reloads the custom diagnostic rule myCustomRules.xml:

**reloadCustomRules(name='myCustomRules.xml')**

## showIncident

Use with WLST: Online

### Description

Shows the details of the specified incident.

### Syntax

```
showIncident(id, [adrHome][, server])
```

| Argument | Definition |
|----------|------------|
| id | The ID of the incident that you want to view. |
| adrHome | The path for the ADR Home from which to query the incident. If this argument is not specified, the default ADR Home will be queried. |
| | The default ADR Home is the following location: |
| | *ADR_BASE*/diag/OFM/*domain_name*/*server_name* |
| server | The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server. |

### Example

The following example displays information about the incident with the ID 10:

```
showIncident(id="10")
Incident Id: 10
Problem Id: 10
Problem Key: MDS-50500 [MANUAL]
Incident Time:Tue May 28 11:02:22 PDT 2013
Error Message Id: MDS-50500
Execution Context:
Flood Controlled: false
Dump Files :
   readme.txt
   jvm_threads10_i1.txt
   dms_metrics11_i1.txt
   dfw_samplingArchive13_i1.JVMThreadDump.txt
   dfw_samplingArchive13_i1.readme.txt
   odl_logs14_i1.txt
   dms_metrics20_i1.txt
```

# Diagnostic Dump Commands

Use the commands in Table 8–3 to display information about dumps and to execute dumps.

*Table 8–3    Diagnostic Dump Commands*

| Use this command... | To... | Use with WLST... |
|---|---|---|
| describeDump | Display a description of the specified diagnostic dump. | Online |
| executeDump | Execute the specified diagnostic dump. | Online |
| listDumps | Display the set of diagnostic dumps that can be executed. | Online |

## describeDump

Use with WLST: Online

### Description

Displays a description of the specified diagnostic dump.

### Syntax

```
describeDump(name [,appName] [.server])
```

| Argument | Definition |
|----------|------------|
| name | The name of the dump for which to display information. |
| appName | The name of the deployed application for which information is gathered. |
| | For example, if you have multiple ADF applications deployed, each may register a dump called adf.dump. To execute this command for a specific application, you must specify the application name. |
| server | The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server. |

### Example

The following example displays information about the dump with the name odl.logs. You use the listDumps command to retrieve the list of available dumps.

```
describeDump(name="odl.logs")
Name: odl.logs
Description: Dump contents of diagnostic logs
Run Mode: asynchronous
Mandatory Arguments:
Optional Arguments:
    Name        Type     Description
    match_all   BOOLEAN  Whether to match both ECID and time range or any one of
them.
    timestamp   LONG     Log message timestamp in milliseconds
    ecid        STRING   Log message execution context ID (ecid)
    exclude_access_logs BOOLEAN  Excludes access logs from dump.
    timerange   LONG     Time range in minutes
```

## executeDump

Use with WLST: Online

### Description

Executes the specified diagnostic dump.

### Syntax

```
executeDump(name [,args] [,outputFile] [,id] [,adrHome] [,server])
```

| Argument | Definition |
| --- | --- |
| name | The name of the diagnostic dump to execute. |
| args | Mandatory or optional arguments to pass to the dump. |
| outputFile | The name of the file to which to write the dump. If you do not specify this argument, the output is written to the console. |
| id | The ID of the incident to which to associate the dump. By default, the specified dump will not be associated with an incident. |
| adrHome | The ADR home that contains the incident. If you do not specify this argument, the default ADR home is used. The default ADR Home is the following location: *ADR_BASE*/diag/OFM/*domain_name*/*server_name* |
| server | The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server. |

Arguments that are either required or are optional can be specified using the "args" keyword. For example:

```
executeDump("java.sysprops",args={"prop" : "os.name"})
```

### Examples

The following example executes the dump with the name jvm.threads and writes it to the file dumpout.txt:

**executeDump(name="jvm.threads", outputFile="/tmp/dumpout.txt")**
```
Diagnostic dump jvm.threads output written to /tmp/dumpoutput.txt
```

The following example executes the dump with the name jvm.threads and the Incident ID for 33 and writes it to the file dumpout.txt:

**executeDump(name="jvm.threads", outputFile="/tmp/dumpout.txt", id="33")**
```
Diagnostic dump jvm.threads output associated with incident 33 in ADR Home
diag/ofm/base_domain/AdminServer
```

The following example executes a dump with the argument prop set to the value os.name:

**executeDump(name="java.sysprops",args={"prop" : "os.name"})**

## listDumps

Use with WLST: Online

### Description

Displays the set of diagnostic dumps that can be executed.

### Syntax

```
listDumps([appName] [,server])
```

| Argument | Definition |
| --- | --- |
| appName | The name of a deployed application for which diagnostics are being gathered. |
| | For example, if you have multiple ADF applications deployed, each may register a dump called adf.dump. To execute this command for a specific application, you must specify the application name. |
| | If you specify this argument, the command returns the dumps for the specified application. If you do not specify this argument, the command returns the system dumps. |
| server | The name of the Managed Server from which to collect information. This argument is valid only when you are connected to the Administration Server. |

### Example

The following example lists all of the available dumps.

```
listDumps()
adf.DiagnosticsJarsVersionDump
dfw.samplingArchive
dms.configuration
dms.ecidctx
dms.metrics
http.requests
jvm.classhistogram
jvm.threads
mds.MDSInstancesDump
odl.activeLogConfig
odl.logs
odl.quicktrace
opss.diagTest
opss.identityStoreUserRoleApiConfig
opss.securityContext
wls.image

Use the command describeDump(name=<dumpName>) for help on a specific dump.
```

# Dump Sampling Commands

Use the commands in Table 8–4 to capture samples of diagnostic dumps at specified intervals.

*Table 8–4    Dump Sampling Commands*

| Use this command... | To... | Use with WLST... |
| --- | --- | --- |
| addDumpSample | Create samplings for Diagnostic Framework dumps. | Online |
| enableDumpSampling | Enable or disables all dump samplings. | Online |
| getSamplingArchives | Collect all dump samplings in a zip file containing the individual sampling files and a readme file. | Online |
| isDumpSamplingEnabled | List whether dump sampling is enabled or disabled. | Online |
| listDumpSamples | List all dump samplings, a specified dump sampling, or all dump samplings associated with a specified server. | Online |
| removeDumpSample | Remove the specified dump sampling. | Online |
| updateDumpSample | Update the specified dump sampling, modifying the settings of the sampling. | Online |

## addDumpSample

Use with WLST: Online

### Description

Creates dump samplings for Diagnostic Framework dumps.

### Syntax

```
addDumpSample(sampleName, diagnosticDumpName [, appName], samplingInterval,
 rotationCount [, dumpedImplicitly] [, toAppend] [, args] [, server])
```

| Argument | Definition |
|---|---|
| sampleName | The name of the sampling. |
| diagnosticDumpName | The name of the diagnostic dump to be sampled. |
| appName | Optional. The name of the application associated with the specified diagnostic dump. If you do not specify appName, the diagnostic dump has a scope of system. |
| samplingInterval | The sampling interval in seconds. If you specify zero or a negative value, sampling is suspended. |
| rotationCount | The maximum number of diagnostic dump samples to be kept in a rotation list. When this limit is reached, the oldest sample is deleted. |
| dumpedImplicitly | Optional. A Boolean value that specifies whether the diagnostic dump archive will be included in the dfw.samplingArchive. Valid values are true and false. The default is true.<br><br>If the value is false, and you want to include the dump archive in the dfw.samplingArchive, you must pass the sampling name to the executeDump command using the args parameter. |
| toAppend | Optional. A Boolean value that specifies whether the diagnostic dump samples are appended to its predecessor, resulting in a single archive when you execute dfw.samplingArchive. Valid values are true and false. The default is true. If the value is true, the sample is appended to its predecessor. If the value is false, dfw.sampleArchive returns a zip file containing individual sample files. Specify false if the dump samples contain binary data. |
| args | Optional. Diagnostic dump arguments to be used by the diagnostic dump at each sampling time. The arguments are expressed as name/value pairs. |
| server | Optional. The name of the server from which to collect the information. If you do not specify this parameter, this command associates the dump sampling with the Administration Server. |

### Example

The following example adds a sampling for the dump dms.metrics:

```
addDumpSample(sampleName='dms_metrics', diagnosticDumpName='dms.metrics',
            samplingInterval=300, rotationCount=10)
```

```
dms_metrics is added
```

## enableDumpSampling

Use with WLST: Online

### Description

Enables or disables all dump samplings. This command affects all configured dump samplings.

### Syntax

```
enableDumpSampling(enable [,server])
```

| Argument | Definition |
| --- | --- |
| enable | A Boolean value that specifies whether to enable or disable dump samplings. Valid values are true and false. |
| server | Optional. The name of the server for which to enable or disable dump sampling. If you do not specify this parameter, this command enables or disables the dump sampling for the Administration Server. |

### Example

The following example disables all dump samplings:

**enableDumpSampling(enable=false)**

```
Dump sampling disabled
```

## getSamplingArchives

Use with WLST: Online

### Description

Collects all dump samplings in a zip file containing the individual sampling files and a readme file. This method is particularly useful in dealing with binary format dumps.

### Syntax

```
getSamplingArchives([sampleName,] outputFile [,server])
```

| Argument | Definition |
| --- | --- |
| name | Optional. The name of a particular dump sampling that you want to retrieve. If you do not specify this argument, the command returns all dump samplings. |
| outputFile | The absolute path of the file to which the dump samplings will be written. |
| server | Optional. The name of the server from which to collect the information. If you do not specify this parameter, this command collects the dump samples for the Administration Server. |

### Example

The following example retrieves the dump sampling for the dump JVMThreadDump:

**getSamplingArchives(sampleName="JVMThreadDump", outputFile="/tmp/jvm_dump.zip")**
```
wrote 63518 bytes to /tmp/jvm_dump.zip
```

The following shows the contents of the zip file:

```
 unzip -l jvm_dump.zip
Archive:  jvm_dump.zip
  Length      Date    Time     Name
 --------      ----    ----     ----
   508780   05-21-13 07:25   dfw_
samplingArchive1065570966467923683.JVMThreadDump.dmp
      840   05-21-13 07:25   dfw_samplingArchive7749640004639161119.readme.txt
 --------                    -------
   509620                    2 files
```

## isDumpSamplingEnabled

Use with WLST: Online

### Description

Lists whether dump sampling is enabled or disabled.

### Syntax

```
isDumpSamplingEnabled([server])
```

| Argument | Definition |
|----------|------------|
| server | Optional. The name of the server to determine if dump sampling is enabled or disabled. This argument is only valid when you are connected to the Administration Server. |

### Example

The following example lists the whether dump sampling is enabled or disabled for the server wls_server_1:

**isDumpSamplingEnabled(server="wls_server_1")**

```
true
```

## listDumpSamples

Use with WLST: Online

### Description

Lists all dump samplings, a specified dump sampling, or all dump samplings associated with a specified server.

### Syntax

```
listDumpSamples([sampleName] [, server])
```

| Argument | Definition |
|---|---|
| sampleName | Optional. The name of the sampling. |
| server | Optional. The name of the server for which to list the dump samplings. If you do not specify this parameter, this command lists the dump samplings for the Administration Server. |

### Example

The following example lists all dump samplings associated with the server wls_server_1:

```
listDumpSamples(server="wls_server_1")
Name              : JVMThreadDump
Dump Name         : jvm.threads
Application Name  :
Sampling Interval : 30
Rotation Count    : 20
Dump Implicitly   : true
Append Samples    : true
Dump Arguments    : context=true, timing=true, progressive=true, depth=20,
threshold=30000

Name              : JavaClassHistogram
Dump Name         : jvm.classhistogram
Application Name  :
Sampling Interval : 1800
Rotation Count    : 5
Dump Implicitly   : false
Append Samples    : true
Dump Arguments    :
```

## removeDumpSample

Use with WLST: Online

### Description

Removes the dump sampling.

### Syntax

```
removeDumpSample(sampleName [,server])
```

| Argument | Definition |
|----------|------------|
| sampleName | The name of the sampling to be removed. |
| server | Optional. The name of the server from which to remove the sampling. If you do not specify this parameter, the dump sampling is removed from the Administration Server. |

### Example

The following example removes the dump sampling named HTTPSampling, associated with the server wls_server_1:

**removeDumpSample(sampleName="HTTPSampling", server="wls_server_1")**

```
Removed HTTPSampling
```

## updateDumpSample

Use with WLST: Online

### Description

Updates the specified dump sampling, modifying the settings of the sampling. You cannot change the name of the sampling. Modifications take affect at the next sampling interval.

### Syntax

```
updateDumpSample(sampleName [, appName], samplingInterval,
    rotationCount [,dumpedImplicitly] [, toAppend] [, arg,]
    [, server])
```

| Argument | Definition |
|---|---|
| sampleName | The name of the dump sampling. |
| appName | Optional. The name of the application associated with the specified diagnostic dump. If you do not specify appName, the diagnostic dump has a scope of system. |
| samplingInterval | Optional. The sampling interval in seconds. If you specify zero or a negative value, sampling is suspended. |
| rotationCount | Optional. The maximum number of diagnostic dump samplings to be kept in a rotation list. When this limit is reached, the oldest sampling is deleted |
| dumpedImplicitly | Optional. A Boolean value that specifies whether the diagnostic dump archive will be included in the dfw.samplingArchive. Valid values are true and false. The default is true. |
| | If the value is false, and you want to include the dump archive in the dfw.samplingArchive, you must pass the sampling name to the executeDump command using the args parameter. |
| toAppend | Optional. A Boolean value that specifies whether the diagnostic dump samples are appended to its predecessor, resulting in a single archive when you execute dfw.samplingArchive. Valid values are true and false. The default is true. If the value is true, the sample is appended to its predecessor. If the value is false, dfw.sampleArchive returns a zip file containing individual sampling files. Specify false if the dump samplings contain binary data. |
| args | Optional. Diagnostic dump arguments to be used by the diagnostic dump at each sampling time. The arguments are expressed as name/value pairs. |
| server | Optional. The name of the server from which to collect the information. If you do not specify this parameter, the dump sampling is updated for the Administration Server. |

### Example

The following example updates the dump sampling HTTPSampling, modifying the sampling interval, rotation count, and server.

```
updateDumpSample(sampleName="HTTPSampling", samplingInterval=200,
                  rotationCount=5, server="wls_server1")

HTTPSampling is updated
```

# 9

# User Messaging Service (UMS) Custom WLST Commands

Oracle User Messaging Service provides a common service responsible for sending out messages from applications to devices. It also routes incoming messages from devices to applications.

This chapter describes the WLST commands that you can use with Oracle User Messaging Service (UMS).

# UMS WLST Command Group

The UMS WLST commands are listed under the command group "ums".

> **Note:** To use these commands, you must invoke WLST from the Oracle home in which the component has been installed. See "Using Custom WLST Commands" in the *Administering Oracle Fusion Middleware.*

## configUserMessagingDriver

Command Category: ums

Use with WLST: Online

### Description

`configUserMessagingDriver` is used to configure messaging drivers.

Specify a base driver type (apns, smpp, email, xmpp, etc.) and a short name for the new driver configuration. The string "usermessagingdriver-" will be prepended to the specified application name.

### Syntax

```
configUserMessagingDriver(baseDriver, appName, driverProperties, clusterName=None,
serverName=None, enabled=true)
```
The use of propertyGroups are deprecated since 12.2.1. Deprecated syntax:

```
configUserMessagingDriver(baseDriver, appName, driverProperties,
clusterName=None
        serverName=None, propertyGroups=None, enabled=true)
```

| Argument | Definition |
|----------|------------|
| baseDriver | Specifies the base messaging driver type. |
| | Must be a known driver type, such as 'apns', 'email', 'extension', 'smpp', 'twitter', or 'xmpp'. |
| appName | A short descriptive name for the deployment. The specified value will be prepended with the string *usermessagingdriver-* |
| driverProperties | An object with the driver properties. It can be an object of the following classes: |
| | CommonDriverProperties, ApnsDriverProperties, EmailDriverProperties, SmppDriverProperties, ExtensionDriverProperties, TwitterDriverProperties, or XmppDriverProperties. |
| | To see all available driver properties for a driver, print the dict field in the class. For example: |
| | print CommonDriverProperties().__dict__ |
| | print EmailDriverProperties().__dict__ |
| serverName | Optional. The name of the managed server for which this configuration shall be valid. One of the clusterName or ServerName should be specified, both are not allowed. If both are None, the configuration becomes managed server level configuration on all managed servers. |
| clusterName | Optional. The name of the cluster for which this configuration shall be valid.One of clusterName or serverName should be specified, both are not allowed.If both are None, the configuration becomes managed server level configuration on all managed servers. |
| enabled | Optional. Specifies if the configuration shall be enabled or disabled. |
| | If not set, default value is true. |

## Examples

### Example 9–1   To configure a XMPP driver with name 'xmpp'

```
driverProperties = XmppDriverProperties()
driverProperties.SenderAddresses = 'IM:alice@example.com'
driverProperties.IMServerHost = 'example.com'
driverProperties.IMServerUsername = 'alice'
driverProperties.IMServerPassword = 'secret'
configUserMessagingDriver(baseDriver='xmpp', appName='xmpp',
    driverProperties=driverProperties, clusterName='my_cluster')
```

### Example 9–2   To configure a Extension driver with name 'extension'

```
driverProperties = ExtensionDriverProperties()
extensionDriverProperties.EndpointURL = 'http://domain.example.com/extension'
extensionDriverProperties.MappedDomain = 'example.com'
extensionDriverProperties.Protocol = 'popup'
configUserMessagingDriver(baseDriver='extension', appName='extension',
    driverProperties=driverProperties)
```

# configUserMessagingServer

Command Category: ums

Use with WLST: Online

## Description

configUserMessagingServer is used to configure the messaging server.

## Syntax

configUserMessagingServer(serverProperties, clusterName=None), serverName=None)

| Argument | Definition |
|---|---|
| serverProperties | An object with the server properties. It must be an object of the class ServerProperties. |
| | To see all available properties, print the dict field in the class. For example: |
| | print serverProperties().__dict__ |
| clusterName | Optional. The name of the managed server for which this configuration shall be valid. One of the clusterName or ServerName should be specified, both are not allowed. If both are None, the configuration becomes managed server level configuration on all managed servers. |
| serverName | Optional. The name of the cluster for which this configuration shall be valid.One of clusterName or serverName should be specified, both are not allowed.If both are None, the configuration becomes managed server level configuration on all managed servers. |

## Examples

***Example 9–3  To configure the JPS Context name for the UMS server(s) in the cluster named 'my_cluster'***

```
serverProperties = ServerProperties()
serverProperties.JpsContext = 'my_jps_context'
configUserMessagingServer(serverProperties=serverProperties, clusterName='my_
cluster')
```

***Example 9–4  To configure the security principal for the UMS server(s) in the domain***

```
serverProperties = ServerProperties()
serverProperties.SecurityPrincipal = 'MyUser'
configUserMessagingServer(serverProperties=serverProperties)
```

## manageUserCommunicationPrefs

Command Category: ums

Use with WLST: Offline

### Description

manageUserCommunicationPrefs is used to download the user messaging preferences from a backend database to the specified XML file, or to upload the user messaging preferences from an XML file into the backend database, or to delete the user preferences from the backend database and backup the preferences to the specified XML file.

### Syntax

```
manageUserCommunicationPrefs (operation={'download' | 'upload' | 'delete'},
filename='file_name', url='jndi_url', username='username', password='password'
 [, encoding='character_encoding'] [, guid='guid1,guid2, ...' ] [, merge={'create_
new' | 'overwrite' | 'append'}] )
```

| Argument | Definition |
|---|---|
| operation | specifies the upload, delete, or download operation to be performed. |
| filename | For download, a unique file name (path) to download the user preferences to. For example, /tmp/download.xml (Linux) or C:\\temp\\download.xml (Windows). |
| | For upload, the file name (path) to upload the user preferences. |
| | For delete, the filename (path) is used to store the removed user preferences. |
| url | The JNDI URL to access the User Messaging Server. For example: t3://<hostname>:<port> |
| username | The user name with login permission to access the User Messaging Server. |
| password | The password of the username. |
| encoding | (Optional) Character encoding to use to download the user preferences. |
| guid | (Optional) The globally unique identifier (guid) of a list of users to use to download their preferences. If no guid is specified, the preferences for all users are downloaded. For delete, the guid specifies the user whose preferences will be removed by this operation. |
| merge | (Optional) This argument is for upload only. Valid values are: |
| | create_new (default): Create new user device, device addresses and/or ruleset entities. An exception will be thrown if an entity with the same primary key already exists and processing will terminate. |
| | overwrite: Remove all existing entities of a user and then create new entities. |
| | append: Only upload entities that do not already exist. |

## Examples

> **Note:** In the URLs below, port 7001 represents the Managed Server port where UMS is deployed.

To download the user messaging preferences of all users to the specified file.

```
wls:offline> manageUserCommunicationPrefs(operation='download',
filename='download.xml', url='t3://localhost:7001', username='weblogic',
password='<password>')
```

To download the user messaging preferences of all users to the specified file using UTF-8 character encoding.

```
wls:offline> manageUserCommunicationPrefs(operation='download',
filename='download.xml', url='t3://localhost:7001', username='weblogic',
password='<password>', encoding='UTF-8')
```

To download the user messaging preferences of the user with guid 'john.doe' to the specified file.

```
wls:offline> manageUserCommunicationPrefs(operation='download',
filename='download.xml', url='t3://localhost:7001', username='weblogic',
password='<password>', guid='john.doe')
```

To download the user messaging preferences of the users with guid 'john.doe' and 'jane.doe' to the specified file using UTF-8 character encoding.

```
wls:offline> manageUserCommunicationPrefs(operation='download',
filename='download.xml', url='t3://localhost:7001', username='weblogic',
password='<password>', guid='john.doe,jane.doe', encoding='UTF-8')
```

To upload the user messaging preferences from the specified file to the backend database.

```
wls:offline> manageUserCommunicationPrefs(operation='upload',
filename='upload.xml',
url='t3://localhost:7001', username='weblogic', password='<password>')
```

To upload the user messaging preferences from the specified file to the backend database and overwrite existing preferences.

```
wls:offline> manageUserCommunicationPrefs(operation='upload',
filename='upload.xml',
url='t3://localhost:8001', username='weblogic', password='<password>',
merge='overwrite')
```

To delete the user preferences of the user with guid 'john.doe' and backup the preferences to the specified file.

```
wls:offline> manageUserCommunicationPrefs(operation='delete',
filename='backup.xml', url='t3://localhost:7001', username='weblogic',
password='<password>', guid='john.doe')
```

To delete the user preferences of the users with guid 'john.doe' and 'jane.doe' and backup the preferences to the specified file using UTF-8 character encoding.

```
wls:offline> manageUserCommunicationPrefs(operation='delete',
filename='backup.xml', url='t3://localhost:7001', username='weblogic',
password='<password>', guid='john.doe,jane.doe', encoding='UTF-8')
```